Complexity Aspects of Map Compression

H. Bodlaender* T. Gonzalez † T. Kloks‡

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

Extended Abstract

Let M be a 2-dimensional colored map which has been digitized into a large 2-dimensional array (M). We define a class of languages (called *rectilinear*) to describe our digitized maps and classify them based on their level of succinct representation. The map compression problem is defined as the problem of finding for any given map a shortest description within a given language. For one dimensional maps, we show that a shortest description can be generated quickly for some languages, but for other languages the problem is NP-hard. We also show that a large number of linear time algorithms for our languages generate map descriptions whose length is at most twice the length of the minimum length description. For all our languages we show that the two dimensional map compression problem is NP-hard. Furthermore, for one of the most succinct of our languages we present evidence that suggests that finding a near-optimal map compression is as difficult as finding an optimal compression.

Let M be a 2-dimensional colored map, e.g., a landscape, to be stored in a digital computer system and/or to be drawn on a terminal screen. Assume that the map has been digitized into a large 2-dimensional array (M). I.e., a large uniform square grid partitions the map into n by m small grid squares denoted by $I_{n,m}$. Grid square $I_{i,j}$ is associated with the matrix entry i,j in M. Each matrix entry ($M(i,j)$) is assigned an integer $l \in [0,p)$ to denote the representative color for grid square $I_{i,j}$.

In many practical applications a map contains large singly colored regions, and also regions in which the colors change rapidly. So, finding a good probabilistic

*This author is partially supported by the ESPRIT II Basic Research Actions Program of the EC, under contract No. 3075 (project ALCOM)

†Current address: Department of Computer Science, University of California, Santa Barbara, CA 93106.

‡This author is supported by the Foundation for Computer Science (S.I.O.N.) of the Netherlands Organization for Scientific Research (N.W.O.).

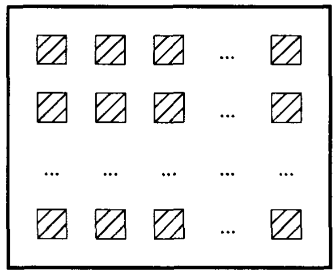

Figure 1: L_{2PDRP} is not as succinct as L_{2RP}

model that represents the distribution of the different colors is at least difficult, if not impossible. Therefore, we shall not make any assumptions about the model that generates our 2-dimensional maps.

In this paper we define a 'language' to describe our digitized maps. The objective is to find a shortest description within the given language. For example, instead of describing a 2-dimensional digitized map by its corresponding matrix we describe it by an *rp-compression*, i.e., sequence of tuples of the form (RP_i, c_i), where RP_i is a subset of grid squares bordered by a simple rectilinear polygon without holes and c_i is a color (i.e., an integer value in the range $[0,p)$). An rp-compression represents map M if map M is generated by coloring all the grid squares in RP_1 with color c_1; then all the ones in RP_2 with color c_2, and so forth. Note that if a grid square is in two or more rectilinear polygons its final color is the last one assigned to it. We shall refer to these rectilinear polygons as *c-rectilinear polygons* and denote the language L_{2RP}. An rp-compression is said to be an *pdrp-compression* if all the c-rectilinear polygons in it are pairwise disjoint. A restricted version of L_{2RP}, which we denote by L_{2PDRP}, is defined by replacing rp-compression by pdrp-compression in the definition of L_{2RP}. We say that the amount of information required to represent a map under L_{2RP} (L_{2PDRP}) is the total number of corners in the c-rectilinear polygons in the rp-compression (pdrp-compression).

Maps usually have a more succinct representation under language L_{2RP} than under language L_{2PDRP}. The following example shows the case when there is a dramatic difference between the minimum length representation of a map in these two languages. The 'map' is given in figure 1. The map consists of $(2n+1)\times(2n+1)$ grid squares, each colored black (represented by a shaded square) or white (represented by a blank). All grid squares are white except for grid squares (i, j) for all i and j even. The smallest description in L_{2PDRP} for the map given in figure 1 contains at least n^2 black grid squares (which are islands in the white area). But, under L_{2RP} the map can be described by one large black square followed by a number of white strips. The amount of data needed to describe our map in language L_{2RP} is only

$O(n)$. Note that a c-rectilinear polygon in an rp-compression does not have to border exactly an area of a given color.

Let M be a map and L be any language. We use $T(L, M)$ to denote the number of 'values' in a minimum length representation M in L. Two languages L and L' are said to be *equivalent* if $T(L, M)$ is $O(T(L', M))$ and $T(L', M)$ is $O(T(L, M))$ for every map M. A language L is said to be *more succinct* than language L' if $T(L, M)$ is $O(T(L', M))$, but $T(L', M)$ is not $O(T(L, M))$, for every map M. One can argue that our classification scheme is not fair because we do not take into account the maximum number of bits in each of the values. So extreme care must be taken when classifying languages because our classification holds only in certain domains. From the above discussion it is simple to show that L_{2RP} is more succinct than L_{2PDRP}. Also, L_{2PDRP} is more succinct than L, where L is the language that represents a map by its n by m matrix of colors. Note that this last comparison is not a fair one because each value in the matrix is an integer value in the range [0,p), where as in the other language the values are integers in the range [0,n). Hereafter we concentrate on languages in which the c-rectilinear polygons may overlap.

An rp-compression is said to be an *rp-nr-compression* if all the c-rectilinear polygons in it assigned the same color are adjacent in the description. A restricted version of L_{2RP} is the language $L_{2RP\text{-}NR}$ obtained by replacing rp-compressions by rp-nr-compressions. The NR stands for *no recoloration* because our procedure that generates M from the rp-nr-compression has the property that once a grid square has been colored with its correct color, it will never be colored with another color different from its correct color. This is not true for rp-compressions. By definition $T(L_{2RP}, M)$ is $O(T(L_{2RP\text{-}NR}, M))$. However, it is not possible to show that $T(L_{2RP\text{-}NR}, M)$ is $O(T(L_{2RP}, M))$ for every map M. Figure 2 shows a class of maps for which this does not hold.

In this paper we study the $2CR_{RP}$ ($2CNR_{RP}$) problem defined as the problem of finding a minimum length representation for a 2-dimensional map in the L_{2RP} ($L_{2RP\text{-}NR}$) language. When the c-rectilinear polygons have exactly four sides (called *c-rectangles*) the above problems are referred to as the $2CR_R$ and the $2CNR_R$ problem. In this case the objective function reduces to minimizing the number of rectangles in the description and we use the term *r-compression* (*r-nr-compression*) instead of rp-compression (rp-nr-compression). The languages are referred to as L_{2R} and $L_{2R\text{-}NR}$, respectively. In each of these two cases the rectangular languages are equivalent (with respect to succinctness) to the rectilinear polygon languages. The reasoning for this is that any rectilinear polygon with k corners may be covered by at most $2k$ rectangles [7] and a rectangle is a rectilinear polygon.

When the two dimensional map has a single row, the map is said to be one dimensional. Voice data files are examples of one dimensional maps. The names for these problems and languages are prefixed by a one instead of a two. Note that in this case the $1CR_{RP}$ ($1CNR_{RP}$) is identical to the $1CR_R$ ($1CNR_R$) problem because all c-rectilinear polygons are simply c-rectangles. All of these one-dimensional languages are equivalent with respect to succinctness. The reason for this is that every r-nr-compression is also an r-compression and thus $T(M, L_{1R})$ is $O(T(M, L_{1R\text{-}NR}))$. The proof of the converse follows from the proof of theorem 3.

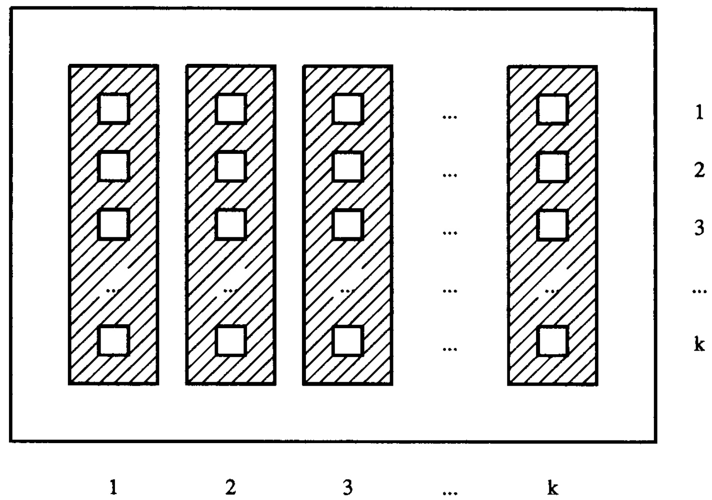

Figure 2: $L_{2RP\text{-}NR}$ is not as succinct as L_{2RP}.

A restricted version of these problems, referred to by $\alpha R\beta_\gamma$, where $\alpha \in \{1,2\}$, $\beta \in \{CR, CNR\}$ and $\gamma \in \{R, RP\}$, is the $\alpha\beta_\gamma$ problem in which each color appears in at most two entries in M. Later on it will be evident why we introduced these versions of our problems.

In many practical situations the number of different colors in M is small. Because of this we shall also investigate the complexity of the $\alpha\beta_\gamma(k)$ problem, where $\alpha \in \{1,2\}$, $\beta \in \{CR, CNR, RCR, RCNR\}$ and $\gamma \in \{R, RP\}$, is the $\alpha\beta_\gamma$ problem in which the number of different colors in M is bounded by the constant k (which is not part of the input).

The One Dimensional Problem

We consider the one dimensional map compression problem with and without recoloration. We present an $O(n^3)$ dynamic programming algorithm for the $1CR_R$, where n is the number of entries in M. However, for the no recoloration variant ($1CNR_R$) we show it is an *NP*-hard problem. For both of these problems we show that any algorithm that avoids a set of "bad decisions" generates a solution within two times the optimal number of c-rectangles in an optimal solution. We also show that for both of these two problems a solution within two times optimal can be generated in linear time. For the $1RCR_R$ and $1RCNR_R$ problems, we present a fast algorithm for its solution by reducing these problems to a well known graph problem which can be solved efficiently.

An instance of our one dimensional problems is represented by $INS = (M_n, p)$, where each $M_i \in [0, p)$. When we refer to a *compression* we mean either an r-compression or an r-nr-compression. Let R_l be a c-rectangle in compression $R = (R_1, R_2, \ldots, R_r)$. We shall refer to the n entries in array M_n as array elements or map grid squares. Element i included in R_l is said to be *tight* if element i is not included in $R_{l+1}, \ldots, R_r$. Since R is a compression, we know that if element i is tight in R_l, then $c_l = M_i$. A c-rectangle is said to be *tight* if the rightmost and leftmost elements in it are tight, and a compression is said to be *tight* if all its c-rectangles are tight.

Algorithm for the $1CR_R$ problem

Let $INS = (M_n, p)$ be any instance of the $1CR_R$ problem. For $1 \leq i \leq j \leq n$, we use $INS_{i,j}$ to represent the subinstance of the $1CR_R$ problem INS defined over array elements $(i, i+1, \ldots, j-1, j)$. Let $g(i,j)$ denote the minimum number of c-rectangles in an optimal solution for the instance $INS_{i,j}$ of the $1CR_R$ problem. Obviously, $g(i,i) = 1$ for $1 \leq i \leq n$. Let $R = (R_1, R_2, \ldots, R_r)$ be an optimal r-compression for $INS_{i,j}$. In lemma 1 we establish an important property of an optimal r-compression for any instance $INS_{i,j}$ of the $1CR_R$ problem. This will aid us in the development of a fast algorithm to find optimal r-compressions .

Lemma 1 *Every instance $INS_{i,j}$ of the $1CR_R$ problem has an optimal r-compression that is tight. Furthermore, element i is tight in c-rectangle R_1.*

Proof:
For brevity the proof is omitted.

□

Let $i_1 < i_2 < \ldots < i_s$ be all the elements in $(i, i+1, \ldots, j-1, j)$ colored M_i. Let R be an optimal r-compression for $INS_{i,j}$ that satisfies the conditions of lemma 1. Let $j_1 < j_2 < \ldots < j_q$ be the tight elements in R_1. From the conditions of lemma 1 we know that $q \geq 1$ and $j_1 = i$. By the principle of optimality it is simple to show that if $q = 1$, then $g(i,j) = g(j_1 + 1, j) + 1$; and if $q > 1$, then $g(i,j) = g(j_1 + 1, j_2 - 1) + g(j_2, j)$, where $g(k,l) = 0$ when $k > l$. Therefore, $g(i,j)$ can be computed via dynamic programming techniques as follows. For $1 \leq i \leq n$, let $g(i,i) = 1$; for $i > j$, let $g(i,j) = 0$; and for $i < j$ define

$$g(i,j) = min\{g(i_1 + 1, j) + 1; min_{1 < k \leq s}\{g(i_1 + 1, i_k - 1) + g(i_k, j)\}\}.$$

We define procedure *DP* to compute the $g(i,j)s$ for all $j - i = 0$, then $1, 2, \ldots$, until $n - 1$, by using the above recursive formulation. It is simple to show that procedure *DP* takes $O(n^3)$ time to find an optimal r-compression for any instance of the $1CR_R$ problem. These results are summarized in the following theorem.

Theorem 1 *Procedure DP generates an optimal r-compression in $O(n^3)$ time for any instance, $INS = (M_n, p)$, of the $1CR_R$.*

Proof:
By the above discussion.

□

Our dynamic programming algorithm cannot be generalized to solve in the same time complexity bound an instance of the $1CNR_R$ problem. The main reason is that now we cannot just introduce a c-rectangle and solve optimally the two remaining subproblems, because for both of these subproblems it is required that the colors of the c-rectangles be in the same order. The following theorem establishes that the $1CNR_R$ problem is NP-hard.

Theorem 2 *The $1CNR_R$ problem is NP-hard.*

Proof:
We prove this theorem by reducing the feedback arc set (FAS) problem ([4]) to the $1CNR_R$ problem. For brevity the proof is omitted.

□

Approximation algorithms for the $1CR_R$ and $1CNR_R$ problems

Let us now consider approximation algorithms for the $1CR_R$ and the $1CNR_R$ problems. We say that an r-compression or an r-nr-compression is ***irreducible*** if no two adjacent elements colored with the same color are tight in different c-rectangles. Given a reducible r-compression or r-nr-compression there is a straight forward procedure to transform it into an irreducible one. The following theorem established the fact that irreducible compressions are good approximations.

Theorem 3 *Any algorithm that generates irreducible r-nr-compressions for an instance INS of the $1CNR_R$ ($1CR_R$) problem with $\hat{f}(INS)$ c-rectangles has the property that $\hat{f}(INS)/f^*(INS) \leq 2$, where $f^*(INS)$ is the number of c-rectangles in an optimal solution for the instance INS of the $1CNR_R$ ($1CR_R$) problem.*

Proof:
The proof is obtained by establishing a lower bound on the number of c-rectangles in an optimal solution. For brevity the details of the proof are omitted.

□

An improved algorithm for the $1RCR_R$ and $1RCNR_R$ problem

It is simple to show that the $1RCR_R$ and the $1RCNR_R$ are identical problems. The dynamic programming algorithm for the $1CR_R$ reduces to an $O(n^2)$ algorithm for $1RCR_R$. We show that in general there exists a somewhat faster algorithm for this case, by reducing our problems to the problem of finding a maximum independent set in an overlap graph [5]. This problem can be solved in $O(dn)$ time [1], where d is the *density*, i.e. the maximum number of intervals including any point.

Theorem 4 *Our algorithm generates an optimal r-compression for the $1RCR_R$ and $1RCNR_R$ problems in $O(dn)$ time, where d is a lower bound on the number of c-rectangles in an optimal solution.*

Proof:
For brevity the proof is not included.

□

The Two Dimensional Problem

We begin by proving that the $2RCR_R$, $2RCNR_R$, $2CR_R$, $2CNR_R$, $2CR_{RP}$ and $2CNR_{RP}$ problems are NP-hard. In many practical cases the number of colors is small compared to the size of the matrix. So the question remains whether a restriction on the number of colors makes the problem computationally simpler. We partially answer this question in the negative. We also show that the problems remain NP-hard even when the number of different colors in M is bounded by a small constant ($2CR_R(4)$ and $2CNR_R(2)$). In addition, we provide evidence that the $2CNR_R(2)$ problem is hard to approximate.

The $2RCR_R$ and related problems

First we show that the $2RCR_R$ problem is NP-hard by reducing a restricted version of the exact cover by three sets ($RXC3$) problem to it. The $RXC3$ problem was shown to be NP-hard in [6]. Then we modify the reduction to establish that the related problems are also NP-hard.

Theorem 5 *The $2RCR_R$, $2RCNR_R$, $2CR_R$, $2CNR_R$, $2CR_{RP}$ and $2CNR_{RP}$ problems are NP-hard.*

Proof:
For brevity the proofs are omitted.

□

We show that the $2CR_R(4)$ problem is NP-hard. We prove this by showing that the FAS problem polynomially reduces to it.

Theorem 6 *The problem $2CR_R(4)$ is NP-hard.*

Proof:
For brevity the proof is omitted.

□

Let us now establish that the $2CNR_R(2)$ problem is NP-hard by reducing the problem of covering a rectilinear polygon with at most k rectangles (CRP) to it. Given a rectilinear polygon RP and integer k, the CRP problem consists of determining whether there is a set of k rectangles that cover RP without covering any point outside RP. The CRP problem was shown to be NP-hard in [2].

Theorem 7 *The $2CNR_R(2)$ problem is NP-hard.*

Proof:
For brevity we do not present details of the proof. However, figure 3 shows the main

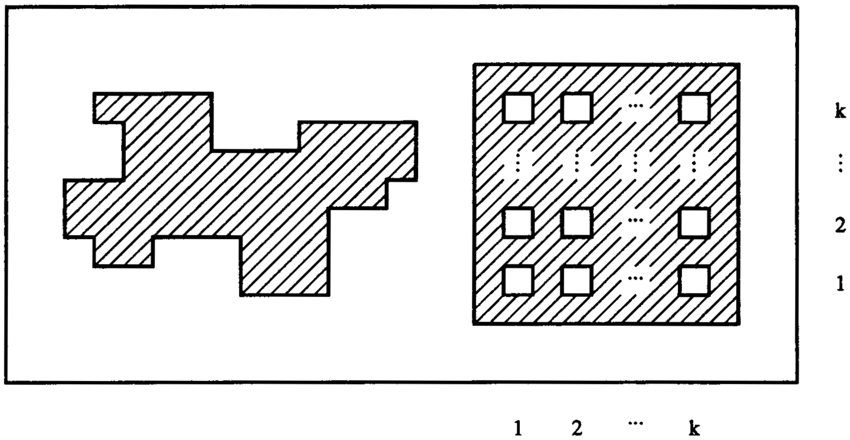

Figure 3: Reduction in theorem 7.

idea behind our reduction.

□

From this reduction and the approximation algorithm for the optimization version of the *CRP* problem given in [7], one may conjecture that there is also an efficient approximation algorithm for the $2CNR_R(2)$ problem. Let us now present evidence to the contrary. The *CRP−WH* problem is the same as the *CRP* problem, except that the new problem has holes (each hole is a rectilinear polygon) inside the rectilinear polygon. The problem consists of covering with k rectangles the rectilinear polygon excluding the exterior and the area covered by the holes. It is simple to show that the *CRP−WH* problem is also an *NP*-complete problem. In what follows when we refer ro the *CRP−WH* problem we mean its corresponding approximation problem. So far, research on developing an efficient approximation algorithm with a constant approximation bound for the *CRP−WH* problem has been fruitless [3]. What we claim is that if there is an efficient approximation algorithm for the $2CNR_R(2)$ problem with approximation bound c, where c is any constant, then there is an efficient approximation algorithm for the *CRP−WH* problem with an approximation bound equal to c', where c' is a constant.

Theorem 8 *The* $2CNR_R(2)$ *approximation problem is as difficult as the CRP−WH problem, i.e., if there is an efficient approximation algorithm for the CRP−WH problem with approximation bound c, where c is any constant, then there is an efficient approximation algorithm for the* $2CNR_R(2)$ *with an approximation bound equal to* c'*, where* c' *is a constant.*

Proof:
For brevity we do not present details of the proof. However, figure 3 (after replacing

the rectilinear polygon by one with holes) shows the main idea behind our reduction.

□

Discussion

We defined a class of languages (called rectilinear) to describe digitized maps and classify them based on their level of succinct representation. For one dimensional maps, we showed that a shortest description can be generated quickly for some languages, but for other languages the problem is *NP*-hard. We also showed that a large number of linear time algorithms for our languages generate map descriptions whose length is at most twice the length of the minimum length description. For all our languages we showed that the two dimensional map compression problem is *NP*-hard. Furthermore, for one of the most succinct of our language we presented evidence that suggests that finding a near-optimal map compression is as difficult as finding an optimal compression.

There are several interesting problems that remain open. The most obvious, is to develop an efficient approximation algorithm for the $2CR_R$, since this involves the most succinct of our languages. Another intersting problem, is to define a new language that is more succinct than the previous ones and for which we can develop efficient exact or approximation algorithms. Perhaps, languages based on primitive objects other than rectangles and rectilinear polygons should be investigated. For example, if the primitive objects are triangles, the resulting languages are more succinct than the rectangular ones. However, if the primitive objects are squares, the resulting languages are not as succinct as the rectangular ones. The two dimesional compression problem with and without recoloration when the primitive objects are triangles can be shown NP-hard by using a reduction similar to the one for theorem 5. For brevity we do not explain this in more detail. There are many ways to view approximations to these problems. A way different to the one explored in this paper, is to relax the restriction that the compression should generate the map exactly. Certainly, such languages would be more succinct than the ones defined in this paper; however it is not clear if shortest descriptions in these languages would be any easier to construct.

In this paper we concentrated in one and two dimensional maps. Another interesting problem, which is as hard as the ones discussed in this paper, are three dimensional maps. This would have applications in terrain data as well as in "movies", which we defined as a sequence of two dimensional maps or frames. Compressions in this case would be important when there is not too much difference between adjacent frames. Perhaps, for certain "movies" even simple heuristics could compress by a significant factor the amount of data needed to store or transmit this information.

References

[1] A. Apostolico and S. Hambrusch, New Clique and Independent Set Algorithms

for Circle Graphs, Technical Report CSD-TR-608, Department of Computer Science, Purdue University, 1986.

[2] J. C. Culberson and R. A. Reckhow, Covering Polygons is Hard, Proceedings of the 29th Annual Symposium on Foundations of Computer Science, 1988, pp. 601-611.

[3] D. S. Franzblau, Performance Guarantees on a Sweep-Line Heuristic for Covering Rectilinear Polygons with Rectangles, *SIAM J. Disc. Math.,* Vol. 2, No. 3, August 1989, pp. 307 - 321.

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* Freeman, San Francisco, 1980.

[5] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs,* Academic Press, New York, 1980.

[6] T. Gonzalez, Clustering to Minimize the Maximum Inter-Cluster Distance, *Theoretical Computer Science,* **38**, October 1985, 293-306.

[7] S.-Y. Wu and S. Sahni, Covering Rectilinear Polygons by Rectangles, *IEEE Transactions on CAD,* Vol. 9, No. 2, April 1990.