Discrete Applied Mathematics 14 (1986) 17-32
North-Holland

THE LINZERTORTE PROBLEM, OR A UNIFIED APPROACH TO PAINTING, BAKING AND WEAVING

Dorit S. HOCHBAUM
School of Business Adm., Univ. of California, 350 Barrows Hall, Berkeley, CA 94720, USA
Edna WIGDERSON
Computer Science Division, Univ. of California, Berkeley, CA 94720, USA

Received 8 April 1985

We present complexity models for measuring the complexity of painting, baking and weaving. In the application to painting our aim is to color the vertices of a bipartite graph according to some proper 2-coloring while using operations that permit the simultaneous coloring of certain subsets of vertices.

We establish lower bounds on the complexity of the painting problem for general 2-colorable graphs (bipartite) and for the special cases of trees and grid graphs. We then describe algorithms that exactly achieve the lower bounds for grid graphs, trees and problems related to baking and weaving.

Keywords. Concrete complexity, coloring, lower bounds, grid graphs, trees, complete bipartite graphs, 2-colorable graphs, Linzertorte problem.

1. Introduction

In this paper we study primarily the complexity of properly painting 2-colorable graphs. Our aim is to achieve a most efficient procedure that will produce such a coloring. In order to do this we must first introduce measures of efficiency for painting algorithms. We describe a general complexity model - the path model that is applicable to all 2-colorable graphs. Then for the class of grid graphs we restrict the general model to three specific models - the vertex model, the line model and the parallel model.

Constructive algorithms that produce proper colorings have been studied in the literature for the problems of vertex and edge coloring. For the problem of vertex coloring there is a constructive and efficient adaptation of Brooks' theorem by Lovász [3]. An example of an efficient algorithm for edge coloring can be found in [2]. The algorithms for vertex and edge coloring all use the basic operation of applying paint to one vertex (or edge) at a time. Our approach allows the simultaneous monochromatic coloring of the vertices of a subgraph that has a prescribed property. Since we study only 2-colorable graphs the problem of identifying the proper color assignment of the vertices is easily resolved, in contrast to the algorithms mentioned above, which must identify the nontrivial proper coloring. This approach is not unique to this paper as it has already been used in [1] and [4] for painting *d*-

dimensional cubes (a special case of grid graphs). Our approach is distinguished from theirs in that we consider different subgraphs (paths as apposed to subcubes or striped subcubes) and in the application motivating the work (baking as opposed to deep issues in complexity theory).

Our models use the simultaneous monochromatic painting of subsets of vertices in the graph where such subsets constitute paths in the graph (any path according to the general model, and paths restricted to a certain family according to the grid models). This improves the efficiency of painting as compared to the straightforward vertex coloring algorithm that uses n steps to paint a graph on n vertices (we assume that the proper color labels of the vertices are available as part of the input). Note that when a path is painted all the vertices along it get the same color, and therefore some of them will be improperly colored. Therefore, their color will have to be corrected in later operations. (As the order of operations determines the resulting coloring, a vertex is assumed to have the last color applied to it).

Our main results include the establishing of an $\Omega(\log_2 n)$ lower bound for the general problem. Moreover, we present an algorithm for painting complete bipartite graphs $K_{s,t}$, where $s+t=n$. The complexity of this algorithm is $O(\log_2 n)$ when s and t are both $\Theta(n)$.

For the class of grid graphs we establish lower bounds for each of the three models and find algorithms with complexities that exactly match those lower bounds.

In the case of trees we prove an $\lceil (n+1)/2 \rceil$ lower bound and present an algorithm with the same complexity.

We also present applications to the domestic arts in general and baking Linzertortes in particular.

The structure of the paper is as follows: we first discuss the case of grid graphs with the application to the Linzertorte problem. The subsequent section is on trees and the last on general bipartite graphs. Section 5 is a summary and discussion of related open problems.

2. Grid graphs

The problem of 2-coloring a d-dimensional grid graph was motivated by the subtleties of baking a Linzertorte cake (see Fig. 1). The reader whose interest in the domestic arts is somewhat limited may feel free to skip the next paragraph.

Baking a Linzertorte involves the creation of an interlaced lattice pattern made from strips of dough (for extra details on the preparation of the cake the reader will find reference [5] rather comprehensive). The dough is soft and tends to tear easily and therefore is difficult to handle. The interlacing involves the delicate operation of lifting the alternate parallel strips that are perpendicular to the strip being placed. We were wondering how to create the weave on top of the cake while doing the least possible number of such operations. Note that if we chose to place first all (say n)

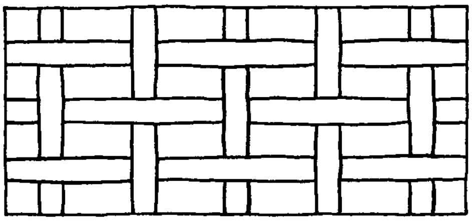


Fig. 1. The Linzertorte cake.

horizontal strips and then all (say m) vertical ones the total number of such operations is $\lfloor nm/2 \rfloor$.

This is also the common method by which weaves are created. Our analysis will show that this naive algorithm can be improved, and we propose the unique optimal algorithm to solve the problem in the smallest possible number of operations.

Note that for the Linzertorte and the weave each point on the lattice has one of two possible positions: (i) horizontal strip on top of a vertical one, or (ii) horizontal strip below a vertical one. Analogously, we can view the problem as that of painting a chessboard. Given an $n \times m$ array of uncolored squares, we apply black or white brush strokes along rows or columns. If the brush is lifted so as to avoid painting some square along the line, then the continuous brush stroke up to this point is considered as an operation.

We are now ready to define the problem in its general context.

A *d-dimensional grid graph* $G^d = (V, E)$ is defined on d sets $N_1, N_2, \ldots, N_d$, where $N_i = \{1, 2, \ldots, n_i\}$ for $i = 1, 2, \ldots, d$, and $V = N_1 \times N_2 \times \ldots \times N_d$. Let $x = (x_1, x_2, \ldots, x_d)$, $y = (y_1, y_2, \ldots, y_d) \in V$. The set of edges is $E = \{(x, y)$: for some i, $|x_i - y_i| = 1$ and $x_j = y_j$ for all $j \neq i\}$.

A grid graph does not contain an odd cycle and hence is 2-colorable. The number of operations of a painting algorithm will be measured according to three different computational models. We will show that the algorithm we propose is optimal with respect to each of the operation models. Indirectly, our analysis also shows that when the n_i's are sufficiently large, the 'lifting' operation discussed above in the context of the Linzertorte and chessboard painting is indeed the most critical operation.

In addition to the applications mentioned, this problem is also related to problems in VLSI design. Particularly, it generalizes the d-dimensional grid graph with $n_1 = n_2 = \ldots = n_d = 2$ (i.e. the d-dimensional unit cube). The coloring of such a graph is discussed in [1] and [4]. Borodin, Dolev, Fich and Paul [1] consider the basic operation of monochromatically painting striped subcubes (a subset of a subcube obtained by specifying the parity of a subset of the components). Plumstead and Plumstead [4] propose an algorithm and a lower bound on the problem complexity both of which are exponential in d. The operation considered in [4] is a monochromatic subcube painting. Note that they use the idea that the first color a node is painted cannot be removed, while we consider the last color applied to remain. By reversing the order of operations the two approaches can be viewed as isomorphic.

2.1. Notation

We will assume from now on that $n_1 \geq n_2 \geq \ldots \geq n_d$. This can always be achieved by relabelling the directions in d-space. We define a d-cube to be a unit cube in d-space.

2.2. The models

We consider two basic operations:

(1) Given i, let x_j be fixed for all $j \neq i$. *Line painting* is the operation of monochromatically painting all vertices in the set

$$\{(x_1, x_2, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots x_d \mid x_i' \textit{ integer } \in [1, n_i]\}$$

(2) *Vertex painting* is the operation of coloring one vertex in V. When we wish to refer to both operations together, we call them interval paintings.

There are three complexity models used based on these operations:

(I) *The vertex model.* This model counts only the vertex painting operations.

(II) *The interval model.* Any interval painting counts as an operation.

(III) *The parallel model.* Painting any number of parallel intervals - each interval monochromatic but possibly different colors applied to different intervals - is considered as one operation. (Two intervals are parallel along direction i if x_i' satisfies $1 \leq l \leq x_i' \leq k \leq n_i$ for both intervals and all other x_j for $j \neq i$ are fixed. Any such fixing of the values of x_j, $j \neq i$ determines another parallel interval). The parallel model, when restricted to the d-cube, is in fact equivalent to coloring subcubes in monochromatic lines. In this sense it is different from both the [1] and the [4] models, where subcubes are monochromatically colored.

2.3. Lower bounds

A d-dimensional grid graph $G^d = (V, E)$ with $V = N_1 \times N_2 \times \ldots \times N_d$ is a collection of $(n_1 - 1)(n_2 - 1) \ldots (n_d - 1)$ d-dimensional unit cubes (referred to as d-cubes). Such a grid graph contains at least $\lfloor n_1/2 \rfloor \lfloor n_2/2 \rfloor \ldots \lfloor n_d/2 \rfloor$ corner disjoint d-cubes. A set of such d-cubes contains all d-cubes of the form

$$[i_1, i_1 + 1] \times [i_2, i_2 + 1] \times \ldots \times [i_d, i_d + 1].$$

where i_j is odd for all j and $1 \leq i_j < n_j$. We denote this set by D. In order to produce a proper coloring for one d-cube in D, one has to perform at least one vertex painting. Otherwise, the two adjacent vertices in the d-cube that were colored by the last line painting end up having the same color. Therefore, each d-cube has at least one improperly colored corner if we apply only line paintings. (An improperly colored vertex may be either uncolored or have the same color as its properly colored neighbors.) In order to properly color these corners, we must apply at least one vertex painting per d-cube in D, since no two adjacent improperly colored corners (in two disjoint d-cubes in D) can be corrected using only one vertex painting. Hence, we have proved the following theorem:

Theorem 1. *The number of vertex painting operations required to color the d-dimensional grid graph with $n_1 n_2 \ldots n_d$ vertices is at least $\lfloor n_1/2 \rfloor \lfloor n_2/2 \rfloor \ldots \lfloor n_d/2 \rfloor$.*

Theorem 1 gives a lower bound for the vertex model. The next theorem implies a lower bound for the interval model and its proof leads to a lower bound for the parallel model.

Theorem 2. *The number of line paintings required to paint a d-dimensional grid graph is at least*

$$\sum_{i=1}^{d} \left\lfloor \frac{n_1}{2} \right\rfloor \cdots \left\lfloor \frac{n_{i-1}}{2} \right\rfloor n_{i+1} \ldots n_d .$$

In conjunction with Theorem 1, *the total number of operations in the interval model is at least*

$$\left\lfloor \frac{n_1}{2} \right\rfloor \cdots \left\lfloor \frac{n_d}{2} \right\rfloor + \sum_{i=1}^{d} \left\lfloor \frac{n_1}{2} \right\rfloor \cdots \left\lfloor \frac{n_{i-1}}{2} \right\rfloor n_{i+1} \ldots n_d .$$

Proof. Each vertex in the graph must be assigned its proper color at least once, by the last operation involving this vertex.

To prove the lower bound we can assume that each vertex gets its proper color exactly once, i.e. if more than one operation involved a certain vertex, all the operations but the last one assigned it the wrong color. This leads to the following minimization problem for the total number of operations:

$$\begin{aligned} &\min\ v + \sum_{i=1}^{d} l_i, \\ &\text{subject to } v + \sum_{i=1}^{d} l_i a_i \geq A, \\ &v \geq \left\lfloor \frac{n_1}{2} \right\rfloor \cdots \left\lfloor \frac{n_d}{2} \right\rfloor, \\ &l_i \geq 0, \quad i = 1, \ldots, d \end{aligned} \tag{1}$$

where

v = the number of vertex paintings;

l_i = the number of line paintings applied in direction i;

a_i = the maximum possible number of vertices painted properly by a line in direction i, i.e. $a_i = \lceil n_i/2 \rceil$; and

A = the number of vertices in the grid graph $= n_1 n_2 \dots n_d$.

Note that as $a_i \geq 1$ for $i = 1 \dots d$, there is always an optimal solution with v held at its lower bound. Therefore, we will assume henceforth that $v = \lfloor n_1/2 \rfloor \dots \lfloor n_d/2 \rfloor$.

Consider a single d-cube. Let H be a graph defined on the 2^d vertices of the d-cube, $V(H)$, and let the edge set of H, $E(H)$, be the edges of the d-cube through which line paintings were applied.

Claim 1. *H is connected.*

Proof. Suppose H has k connected components and consider the last line painting applied in each component. This will surerly leave at least one vertex in the component improperly colored, and as no more line paintings will be applied to edges in this component, the vertex will have to achieve its proper coloring by vertex painting. By the assumption on the value of v and theorem 1 there is exactly one vertex painted by a vertex painting operation in every d-cube, and therefore H must have exactly one connected component.

As H is connected, $|E(H)| \geq |V(H)| - 1 = 2^d - 1$. Also in each d-cube there exists an edge in $E(H)$ which was painted by a line painting in each direction i, $i = 1, 2, \dots, d$, otherwise H will not be connected. Therefore l_i, the number of lines painted in direction i in the grid graph is bounded below by the number of d-cubes projected on the hyper-plane orthogonal to direction i, or

$$l_i \geq \left\lfloor \frac{n_1}{2} \right\rfloor \cdots \left\lfloor \frac{n_{i-1}}{2} \right\rfloor \left\lfloor \frac{n_{i+1}}{2} \right\rfloor \cdots \left\lfloor \frac{n_d}{2} \right\rfloor \equiv b_i^{(d)}.$$

Noting that

$$A - v = n_1 \dots n_d - \left\lfloor \frac{n_1}{2} \right\rfloor \cdots \left\lfloor \frac{n_d}{2} \right\rfloor \geq (2^d - 1) \left\lfloor \frac{n_1}{2} \right\rfloor \cdots \left\lfloor \frac{n_d}{2} \right\rfloor \equiv B_d$$

we derive a new minimization problem, whose solution value is a lower bound to the number of line paintings:

$$\begin{aligned} &\min \sum_{i=1}^{d} l_i^{(d)}, \\ &\text{subject to } \sum_{i=1}^{d} a_i l_i^{(d)} \geq B_d, \\ &l_i^{(d)} \geq b_i^{(d)}. \end{aligned} \tag{2_d}$$

Claim 2. *There is an optimal solution to problem* (2) *with* $l_d^{(d)} = b_d^{(d)}$.

Proof. Let $\{l_1', \dots, l_d'\}$ be an optimal solution to (2_d) with $l_d' = b_d^{(d)} + \varepsilon$, $\varepsilon > 0$. Define a new solution: $l_1 = l_1' + \varepsilon$, $l_i = l_i'$ for $2 \le i \le d-1$, and $l_d' = l_d - \varepsilon = b_d^{(d)}$. Taking the fact $a_1 \ge a_2 \ge \dots \ge a_d$ into account, clearly $\{l_1, l_2, \dots, l_d\}$ is a feasible solution, and its value is less than the optimum, contradiction.

Combining the results of Claims 1 and 2, we conclude that for every d-cube there is exactly one edge corresponding to a line painting in direction d. Fixing the lines painted in direction d gives n_d minimization problems in the $(d-1)$-dimensional space of the form (2_{d-1}). Repeating Claim 2 for the problems (2_{d-1}) we find that each has an optimal solution with

$$l_{d-1}^{(d-1)} = b_{d-1}^{(d-1)} = \left\lfloor \frac{n_1}{2} \right\rfloor \left\lfloor \frac{n_2}{2} \right\rfloor \cdots \left\lfloor \frac{n_{d-2}}{2} \right\rfloor .$$

Iterating this procedure, we get at step i, $n_{d-i+1} n_{d-i+2} \cdots n_d$ problems of the form (2_{d-i}), and each has an optimal solution with

$$l_{d-i}^{(d-i)} = b_{d-i}^{(d-i)} = \left\lfloor \frac{n_1}{2} \right\rfloor \left\lfloor \frac{n_2}{2} \right\rfloor \cdots \left\lfloor \frac{n_{d-i-1}}{2} \right\rfloor \quad \text{for } i = 0, \dots, d-1.$$

Summing all the l_i's up, gives the required result and completes the proof of Theorem 2. □

Corollary 1. *If $n_i = O(n)$ for $i = 1, 2, \dots, d$, then a lower bound for the total number of line paintings is $O(n^{d-1})$ and a lower bound on the complexity of the problem given by the line model is $O(n^{d-1}) + O(n^d) = O(n^d)$.*

Corollary 2. *If $n_i = O(1)$ for $i = 1, 2, \dots, d$, then a lower bound on the total number of operations is exponential in d. If $n_i = 2$ all i, then we get $2^d - 1$ line paintings plus one vertex painting operation. In this case the algorithm offers no improvement compared to the naive algorithm that paints one vertex at a time.*

Corollary 3. *A lower bound for the total number of line paintings for the parallel model is $d + \lfloor n_d/2 \rfloor$.*

Proof. By Claim 1 in Theorem 2, we must have lines in each direction, and we also need at least $\lfloor n_d/2 \rfloor$ vertex paintings in direction d to recolor improperly colored vertices.

Remark. Another operation which might be considered is a *strict interval painting* operation. This operation is the monochromatic painting of an interval in direction i which is of length greater than 1 and less than n_i. Introducing an additional type of operation can only reduce the total number of operations. This is indeed the case when we allow vertex, line and strict interval paintings. In the d-dimensional case the lower bound is reduced (for n_1, $n_2 \ge 3$)

$$\text{from} \quad \left\lfloor \frac{n_1}{2} \right\rfloor \left\lfloor \frac{n_2}{2} \right\rfloor + n_2 + \left\lfloor \frac{n_1}{2} \right\rfloor$$

$$\text{to} \quad \left\lfloor \frac{n_1}{2} \right\rfloor \left\lfloor \frac{n_2}{2} \right\rfloor + \left\lceil \frac{n_2}{2} \right\rceil + \left\lceil \frac{n_1}{2} \right\rceil + 1.$$

This lower bound is also matched by an optimal algorithm (that uses exactly the lower bound number of operations). The details of both the lower and upper bound proofs, which are due to Howard Karloff, are omitted.

We next present an algorithm that matches precisely the lower bound.

2.4. *The coloring algorithm*

The algorithm we present achieves precisely the lower bound for all three models. It is therefore an optimal algorithm.

We first illustrate the application of the algorithm for a 7×7 chessboard (we chose it over the 8×8 board since this allows us to illustrate the role of rounding down $\lfloor n_i/2 \rfloor$ in the algorithm).

Example 1 (7×7 chessboard). Paint alternating horizontal white and black lines ((a)). Paint black vertical lines ((b)) in the 2nd, 4th, 6th positions. Finally ((c)) correct with white interval paintings the appropriate positions along the black vertical lines. See Fig. 2.

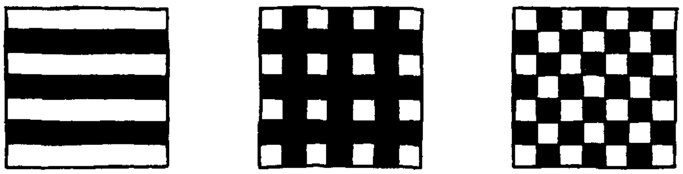
This gives us the following numbers of operations:

(I) according to the interval model $9 = \lfloor 7/2 \rfloor \lfloor 7/2 \rfloor$ operations,

(II) according to the line model $19 = 7 + \lfloor 7/2 \rfloor + \lfloor 7/2 \rfloor \lfloor 7/2 \rfloor$ operations,

(III) according to the parallel model $5 = 2 + \lfloor 7/2 \rfloor$ operations (2 parallel line paintings and 3 parallel vertex paintings).

Example 2 (8×5 Linzertorte). The Linzertorte problem is a bit more complex since the order of placing the dough strips determines the 'color'. Consider an 8×5 example: First place horizontal strips in the even positions (2nd, 4th, 6th and 8th). Then



7 horizontal line paintings (a)

3 vertical line paintings (black) (b)

9 strict interval paintings (white) (c)

Fig. 2.

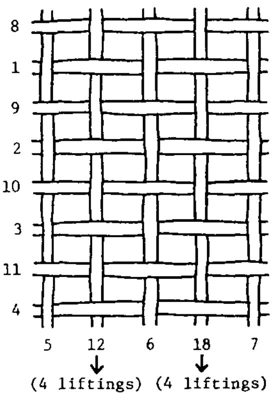


Fig. 3.

place vertical strips in odd positions (1st, 3rd, 5th). Now place the remaining horizontal strips in the odd positions. Finally, place the even vertical strips (2nd, 4th), using the lifting operation (vertex painting) in the 2nd, 4th, 6th, 8th horizontal positions. As a result we get a weave involving only $8=\lfloor 8/2\rfloor \lfloor 5/2\rfloor$ lifting operations and the equivalent of 11 line operations. This process is illustrated in Fig. 3.

We denote a line along the first dimension with the $d-1$ other coordinates fixed at $x_2, x_3, \ldots, x_d$ $(1 \leq x_i \leq n_i)$ by $(-, x_2, x_3, \ldots, x_d)$. To indicate a line where some of the coordinates can be fixed at odd or even values only, we affix the words odd or even respectively to these coordinates (e.g., if the line is in direction 1 and x_2 must be odd and x_7 must be even we write ($-$, x_2 odd, $x_3, \ldots, x_7$ even, $\ldots, x_d$)). Recall that $n_1 \geq n_2 \geq \ldots \geq n_d$.

The Coloring Algorithm

Step 1. Paint all lines $(-, x_2, x_3, \ldots, x_d)$ black if $x_2+x_3+\ldots+x_d$ is even, white if $x_2+x_3+\ldots+x_d$ is odd.

Step 2. Paint all lines (x_1 even, $-, x_3, x_4, \ldots, x_d$) black if $x_3+x_4+\ldots x_d$ is even, white if $x_3+x_4+\ldots x_d$ is odd.

Step 3. Paint all lines (x_1 even, x_2 even, $-, x_4, \ldots, x_d$) black if $x_4+x_5+\ldots+x_d$ is even, white if $x_4+x_5+\ldots+x_d$ is odd. And in general, for $i=4, 5, \ldots d-1$

Step i. Paint all lines (x_1 even, x_2 even, $\ldots, x_{i-1}$ even, $-, x_{i+1}, \ldots, x_d$) black if $x_{i+1}+x_{i+2}+\ldots+x_d$ is even, white if $x_{i+1}+x_{i+2}+\ldots+x_d$ is odd.

Step d. Paint all lines (x_1 even, $\ldots, x_{d-1}$ even, $-$) black.

Step d+1. Apply white vertex paintings to all improperly colored vertices along the black lines painted in Step *d*.

Complexity of each step

Step 1: $n_2 n_3 \ldots n_d$ line paintings.

Step 2: $\lfloor n_1/2\rfloor n_3 n_4 \ldots n_d$ line paintings.

Step i: $\lfloor n_1/2\rfloor \lfloor n_2/2\rfloor \ldots \lfloor n_{i-1}/2\rfloor n_{i+1} \ldots n_d$ line paintings.
Step d: $\lfloor n_1/2\rfloor \lfloor n_2/2\rfloor \ldots \lfloor n_{d-1}/2\rfloor$ line paintings.
Step d+1: $\lfloor n_1/2\rfloor \lfloor n_2/2\rfloor \ldots \lfloor n_d/2\rfloor$ vertex paintings.
The complexity of the algorithm is therefore equal to the lower bound and is of order $O(\prod_{i=1}^{d} \lfloor n_i/2\rfloor)$. That is, the vertex paintings are the dominant operation.

The validity of the algorithm. It suffices to verify that when the painting algorithm terminates all vertices of the d-dimensional grid graph are properly colored, i.e., all vertices of even parity are colored white, and all vertices of odd parity are colored black. Note that in each step of the algorithm more vertices are added to the set of property colored vertices, but none are removed from that set.

After Step 1, all vertices with x_1 odd become properly colored.

After Step 2, all vertices with x_1 even and x_2 odd become properly colored.

After Step i, all vertices with $x_1, \ldots, x_{i-1}$ even and x_i odd become properly colored ($1 < i \leq d$).

After Step d all vertices with $x_1, \ldots, x_{d-1}$ even and x_d odd become properly colored. The remaining vertices that are still improperly colored at the end of Step d are precisely the vertices with all coordinates even. These vertices are finally properly colored using the interval paintings of Step $d+1$.

Remarks. In the $n \times m$ two-dimensional case (e.g. the Linzertorte) our algorithm improves on the common weaving algorithm discussed above by a factor of 2: it reduces the number of lifting operations (vertex paintings) from $\lfloor nm/2\rfloor$ to $\lfloor n/2\rfloor \lfloor m/2\rfloor$. It is interesting to observe that for the Linzertorte or the weaving problem it is impossible to use more than $nm - \lfloor n/2\rfloor \lfloor m/2\rfloor$ such operations. This is because in this case all nm horizontal and vertical lines must be placed, and for a 2-cube, after the first line has been placed (this cannot involve a lifting operation), we can execute at most 3 lifting operations, one for each line placed. Therefore the naive algorithm constitutes an average between the worst and the best possible algorithmic complexities for this case.

Each step of the parallel algorithm can be interpreted as a generalized subcube painting. It might seem therefore that our algorithm is better than the known $\{0,1\}^d$ (the d-cube) painting algorithm. This is however not be case because our generalized subcube can not be painted monochromatically. Only the individual lines are monochromatic.

3. Trees

Grid graphs are a special class of 2-colorable graphs. Another class of 2-colorable (bipartite) graphs are trees. A natural extension of the line or interval painting operation is a monochromatic path painting operation. We now establish a lower bound on the complexity of painting a tree using path painting operations. Then we present an algorithm with complexity matching the lower bound exactly.

We view any graph coloring algorithm A as a sequence of operations $((p_1, x_1), (p_2, x_2), \ldots, (p_k, x_k))$ where $p_i = (v_{i_1}, v_{i_2}, \ldots, v_{i_{l_i}})$, for some integer $l_i > 1$, is a path in the graph, i.e. $v_{i_j} \in V$ for all $j = 1, 2, \ldots, l_i$, and $(v_{i_j}, v_{i_{j+1}}) \in E$ for all $j = 1, 2, \ldots, L_i - 1$, and x_i is the color with which the path p_i is painted. When $l_i = 1$ we refer to the operation as vertex painting. With every p_i, we associate the set of vertices on it, and denote it by $p_i' = \{v_{i_1}, v_{i_2}, \ldots, v_{i_{l_i}}\}$. We define the algorithm A to be in canonical form, if all vertex painting operations occur as soon as possible. Formally, if $|p_i'| > 1$ and for some $j \geq 1$, $|p_{i+j}'| = 1$ and $p_{i+l}' \cap p_{i+j}' = \emptyset$ for $l = 0, 1, \ldots j-1$, then insert (p_{i+j}, x_{i+j}) prior to (p_i, x_i) in the sequence. Also, the canonical form of an algorithm does not include more than one path painting operation per path. Further, we will assume that all paths painted have an odd number of vertices, and the color of the path is the proper coloring of the endpoints. To see that this is a valid assumption, note that if a path with an even number of vertices is painted, then one of its end-points will get the wrong color, and will have to be corrected. Painting the path without this end-point and dealing with it seperately won't change the total number of coloring steps. We denote by $|A|$ the number of pairs (p_i, x_i) in the sequence which describes A, and define $|A|$ to be the complexity of A.

Clearly, many different algorithms can be transformed to the same canonical form, the complexity of which bounds from below the complexity of all algorithms from which it can be derived. Hence, it will suffice to consider only algorithms in canonical form for lower bound derivations.

Definition. An algorithm A in canonical form *w-semi-properly* colors a tree $T = (V, E)$ with $w \in V$ if all the components of T-$\{w\}$ are each properly colored by A, and the last path p with $|p'| > 1$ colored by A contains w.

Note. This coloring need not be consistent with any coloring of T.

Theorem 4. *Any algorithm which colors a tree $T = (V, E)$ on n nodes w-semi-properly uses at least $\lfloor n/2 \rfloor$ painting operations.*

Proof. By induction on $n : n = 1$ is a trivial case.

Assume the theorem holds for all trees on less than n vertices. Let $T = (V, E)$ be a tree on n vertices, and A an algorithm which w-semi-properly colors it for some $w \in V$. Let p, $|p'| = k > 1$ be the last (non-empty) path colored by A ($w \in p$ by definition). Shrink p to a single vertex $\bar{w}$, and define a new tree $\bar{T} = (\bar{V}, \bar{E})$, where $\bar{V} = V$-$p' \cup \bar{w}$ ($|\bar{V}| = n - k + 1$) and $\bar{E} = \{(u, v) \in E \mid u, v \notin p\} \cup \{(u, \bar{w}) \mid \exists v \in p \text{ such that } (u, v) \in E\}$. We can define a 1-1 mapping, φ, from paths in T onto paths in $\bar{T}$ as follows: if a path in T did not intersect p, then it is mapped by the identity mapping, and if a path $q = (q_1, q_2, \ldots, q_l)$ has vertices $q_i, q_{i+1}, \ldots, q_m \in p$ ($1 \leq i \leq m \leq l$ and the intersection vertices must be consecutive on both paths as they are in a tree), then it is mapped to the path $\bar{q} = (q_1, q_2, \ldots, q_{i-1}, \bar{w}, q_{m+1}, \ldots, q_l)$. A induces an algorithm $\bar{A}$ which $\bar{w}$-semi-properly colors $\bar{T}$. In fact, $\bar{A}$ is A truncated just before the step which paints p with every prior path q replaced by $\varphi(q)$. Conversely, A can be viewed as $\bar{A}$ where every path q is replaced by $\varphi^{-1}(q)$ followed by an algorithm B which w-semi-properly colors the path p.

By the induction hypothesis, $|\bar{A}| \geq \lfloor (n-k+1)/2 \rfloor$ and $|B| \geq \lfloor k/2 \rfloor$, therefore

$$|A| \geq \left\lfloor \frac{n-k+1}{2} \right\rfloor + \left\lfloor \frac{k}{2} \right\rfloor \geq \left\lfloor \frac{n}{2} \right\rfloor . \qquad \square$$

Corollary. *Any algorithm which colors a tree on n vertices must use at least $\lfloor n/2 \rfloor + 1$ coloring operations.*

Proof. Let $A = ((p_1, x_1), (p_2, x_2), \ldots, (p_k, x_k))$ be an algorithm which colors a tree on n vertices, T. Clearly, $|p_k'| = 1$, as otherwise all the vertices on the path would have the same color and the coloring would be improper. Therefore, $A' = ((p_1, x_1), \ldots, (p_{k-1}, x_{k-1}))$ is a p_k-semi-proper coloring algorithm for T.

$$|A| = |A'| + 1 \geq \left\lfloor \frac{n}{2} \right\rfloor + 1. \qquad \square$$

Note that this proof does not hold for general graphs as the mapping between paths in the original and new trees is not necessarily a mapping of paths to paths in a general graph. This happens when paths can intersect the last path painted on non-consecutive vertices.

For the purpose of the tree-coloring algorithm, we assume that the vertices of the input tree (T) are labelled w or b according to the color they will eventually assume. An even path is defined here to be a path with an even number of vertices. $V(T)$ denotes the tree's set of vertices.

```
Tree-coloring algorithm($T$)
begin
0: while $T \neq$ an even path or a single vertex do
   begin
      find 2 leaves $s, t \in T$ that have the same label and paint the path connecting $s$
      and $t$ in $T$ with the color indicated by their label
      $T := $ the tree induced on $V(T) - \{s\} - \{t\}$
   end
   if $T =$ an even path then
   begin
      paint one end point, $v$, according to its label
      $T := $ the tree induced on $V(T) - \{v\}$ go to 0
   end
   if $|V(T)| = 1$ paint the vertex according to its label
end
```

Theorem 5. *The complexity of the tree coloring algorithm for coloring a tree T on n vertices is $\lfloor n/2 \rfloor + 1$.*

Proof. As long as the 'while' loop in the algorithm is executed, every operation will reduce the number of nodes in the tree by two. The loop is exited the first time on one of the following conditions;

(i) The tree is reduced to an path with an even number, k, of vertices.

(ii) The tree is reduced to a single vertex.

Case (i) implies that originally the tree had an even number of vertices. After one end point is properly colored, the 'while' loop is entered again, and exited just once more, when the tree has been further reduced to a single vertex. The total number of operations is:

$$\frac{n-k}{2} + 1 + \left\lfloor \frac{k-1}{2} \right\rfloor + 1 = \frac{n-2}{2} + 2 + \frac{n}{2} + 1 = \left\lfloor \frac{n}{2} \right\rfloor + 1.$$

Case (ii) implies that originally the tree had an odd number of vertices, and the total number of painting operations is: $(n-1)/2 + 1 = \lfloor n/2 \rfloor + 1$. □

4. General 2-colorable graphs

For general 2-colorable graphs we use the same path painting model as described for trees. Here we prove a lower bound on the complexity of painting the graphs, and present the fastest painting algorithm we could find.

Theorem 6. *For any bipartite graph $G=(V, E)$ any coloring algorithm will use at least $\lceil \log_2 n \rceil$ steps, where $n = |V|$.*

Proof. Let $X^{(i)}$ be the set of improperly colored vertices after Step i in the algorithm, and let $|X^{(0)}| = n$. Let Step $i+1$ be the coloring of a path p, with set of vertices p'. Define $m = |X^{(i)} \cap p'|$. Clearly, $|X^{(i)}| \geq m$ and $|p'| \geq m$. After p has been colored, at least $\lfloor |p'|/2 \rfloor$ vertices on it are improperly colored, so

$$|X^{(i+1)}| \geq |X^{(i)}| + \left\lfloor \frac{|p'|}{2} \right\rfloor - m \geq |X^{(i)}| + \left\lfloor \frac{m}{2} \right\rfloor - m = |X^{(i)}| - \left\lceil \frac{m}{2} \right\rceil$$
$$\geq |X^{(i)}| - \left\lceil \frac{|X^{(i)}|}{2} \right\rceil = \left\lfloor \frac{|X^{(i)}|}{2} \right\rfloor,$$

which shows that at least $\lceil \log_2 n \rceil$ steps are needed to properly color the graph. □

If the lower bound can ever be achieved, it must be achieved on complete bipartite graphs. This is because the painting complexity cannot increase with adding edges to the graph. We actually know that the lower bound of $\lceil \log_2 n \rceil$ is not achievable.

Dick Karp suggested a simple dynamic programming algorithm (for which each entry in the dynamic programming table is a pair describing the number of vertices that are finally and properly painted on each side of the bipartition) that evaluates the optimal algorithm and minimum number of operations for each bipartite graph $K_{n,m}$. As it turns out, the lower bound is exceeded even for complete bipartite graphs of the form $K_{n,n}$. The algorithm, however, does not provide a clue as to a closed form expression for the lower bound.

We now present an $O(\log_2 n)$ painting algorithm for the complete bipartite graphs on n vertices $K_{n/2,n/2}$ (the complexity of which differs by a factor of 2 from the lower bound).

The idea of the algorithm is to paint at each iteration a graph of the form $K_{m,m}$ with a subgraph $K_{\lceil m/2\rceil,\lceil m/2\rceil}$ properly painted using *two* operations, and the remainder of the graph $K_{\lfloor m/2\rfloor,\lfloor m/2\rfloor}$ is left improperly colored. Eventually, the improperly colored graph is reduced by the algorithm to one of the bipartite graphs $K_{1,1}$, $K_{2,2}$, $K_{3,3}$ or $K_{4,4}$ which can be painted using 2,3,4,4 operations respectively. The complexity of the algorithm is therefore $2\lceil\log_2 n\rceil-2$ (where $n=m+m$).

Let the labels of vertices in one set of the bipartition be the odd numbers in $\{1,2,\dots,2m\}$ and in the other the even numbers. The first (odd) set will be painted black and the other white.

```
**Algorithm for painting** $K_{m,m}$ (due to M. Saks)
begin
  while $m \geq 5$ do {paint $K_{\lceil m/2\rceil,\lceil m/2\rceil}$ properly}
  begin
    paint the path $(1,2,3,\dots,2m)$ black
    paint the path $(1,2\lfloor m/2\rfloor+2,3,2\lfloor m/2\rfloor+4,\dots,2m)$ white
    $m:=\lfloor m/2\rfloor$
  end
  paint $K_{m,m}$
end
```

By a trivial modification this algorithm paints also $K_{m,m+1}$, and the complexity remains $2\lceil\log_2 n\rceil-2$, where $n=2m+1$.

It is easy to derive a similar algorithm for $K_{s,t}$ where $s<t$ by partitioning the t vertices into blocks of size $s+1$. This gives a complexity measure of $O(t/s)$, which can be $O(n)$ $(n=s+t)$in the worst case. An example for this is the painting of $K_{1,n-1}$ according to the tree painting algorithm. Another interesting case of complete bipartite graphs is when the graph is if the form K_{f_{l+1},f_l} where f_l and f_{l+1} are consecutive Fibonacci numbers. This algorithm uses l path paintings exactly, or $\log_{1.6} n-2$, where $n=f_{l+1}+f_l$. An algorithm for painting these graphs is as follows: follows:

Let the labels of the vertices in the set of size f_{l+1} be $1,2,\dots,f_{l+1}$, and in the other set be $1',2',\dots,f_l'$. The colors used will be 0 and 1.

Algorithm for painting $K_{f_{l+1}f_l}$ (due to N. Allon)

```
begin
  col: = 0
  while $f_l > 0$ do
  begin
    paint the path $(f_{l-1}+1, 1', f_{l-1}+2, 2', \ldots, f_l'-1, f_{l-1}+f_l)$ col
    col: = 1-col
    $l := l-1$
    relabel $i$ by $i'$
    relabel $i'$ by $i$
  end
  paint the remaining vertex col
end
```

For incomplete bipartite graphs the algorithm could be as bad as linear in the number of vertices. Consider for instance pyramid graphs of width (or depth) m. A straightforward algorithm paints such graphs properly using m operations, whereas a lower bound of m steps has been established for m odd (by Howard Karloff and Dani Soroker).

5. Summary and open problems

We have dealt with the problem of properly coloring 2-colorable graphs using the atomic operation of monochromatically painting paths (which in certain cases were restricted to be taken out of a specified set of paths). There are a lot of extensions and related problems which arise from this and we will conclude by presenting a few:

(I) Coloring a graph which is known to be k-colorable for some $k>2$. Clearly, this is much more difficult than in our case, especially as there may be more than one legal coloring for such graphs.

(II) Coloring a k-colorable graph for $k \geq 2$ where at each step one can color monochromatically subsets of the graph taken from a specified set of graphs (e.g. trees, stars, etc.).

(III) Actually, there is no need to restrict the problem to legal colorings, so we can phrase the problem as follows: Given 2 copies of a graph, where one of them has colors assigned to its vertices, what is the smallest number of monochromatic operations needed to assign the same colors to the corresponding vertices of the other graph. Again, at each step one may monochromatically color the vertices of a subgraph taken from a certain family of graphs. We suspect that if the graph is to be colored illegally with its chromatic number of colors, the complexity will not exceed that of the legal coloring.

Acknowledgements

We would like to express our gratitude to David Shmoys for bringing reference [5] to our attention, to Avi Wigderson for reference [1] and valuable suggestions, to Mike Saks for the algorithm for painting complete bipartite graphs of equal sets; to Noga Alon for painting complete bipartite graphs with consecutive Fibonacci numbers; to Dick Karp for the dynamic program algorithm that evaluates the minimal number of operations in a complete bipartite graph; to Howard Karloff for his lower and upper bound proof for the grid graphs with strict interval painting operations; to Ilan Adler for asking all the wonderfully naive questions, and to all for their interest in the problem. David, Avi, Aharon Hochbaum and the BATS crowd should also be noted for the apparent appetite and enthusiasm with which they consumed our preliminary studies of the Linzertorte problem.

References

[1] A. Borodin, D. Dolev, F. Fich and W. Paul, Bounds for width-two branching, programs, Proc. 15th ACM Symp. on Theory of Computing (1983) 87-93.

[2] D.S. Hochbaum, T. Nishizeki and D.B. Shmoys, A better than best possible algorithm to edge-color multigraphs, J. Algorithms, to appear.

[3] L. Lovász, Three short proofs in graph theory, J. Combin. Theory (B) 19 (1975) 269-271.

[4] B.R. Plumstead and J.B. Plumstead, Bounds for cube coloring, SIAM J. Algebraic Discrete Methods 6 (1) (1985).

[5] J. Wechsburg, The Cooking of Vienna's Empire (Time-Life Books, New-York, 1968) 191-192.