Algorithms for longer OLED Lifetime

Friedrich Eisenbrand[1], Andreas Karrenbauer[2], and Chihao Xu[3]

[1] Fachbereich Mathematik, Universität Paderborn, eisen@math.uni-paderborn.de
[2] Max-Planck-Institut für Informatik, Saarbrücken, karrenba@mpi-inf.mpg.de
[3] Lehrstuhl für Mikroelektronik, Universität Saarbrücken,
chihao.xu@lme.uni-saarland.de

Abstract. We consider an optimization problem arising in the design of controllers for OLED displays. Our objective is to minimize the amplitude of the electrical current flowing through the diodes which has a direct impact on the lifetime of such a display. The optimization problem consist of finding a decomposition of an image into subframes with special structural properties that allow the display driver to lower the stress on the diodes. For monochrome images, we present an algorithm that finds an optimal solution of this problem in quadratic time. Since we have to find a good solution in realtime, we consider an online version of the problem in which we have to take a decision for one row based on a constant number of rows in the lookahead. In this framework this algorithm has a tight competitive ratio. A generalization of this algorithm computes near optimal solutions of real-world instances in realtime.

1 Introduction

Organic Light Emitting Diodes (OLEDs) have received growing interest recently as more and more commercial products are equipped with such displays. Though they have many advantages over current technology like LCD, only small size OLED displays have entered the marked yet. One reason for this is the limited lifetime of those

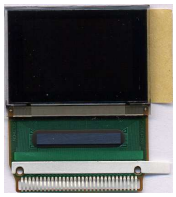

Fig. 1. Sample of a commercial OLED device with integrated driver chip

displays. While a lot of research is conducted on the material science side, the so-called *Multiline Addressing Scheme* for passive matrix OLED displays [7] tackles the lifetime-problem from an algorithmic point of view. It is based on the fact that equal rows can be displayed simultaneously with a lower electrical current than in a serial manner. An

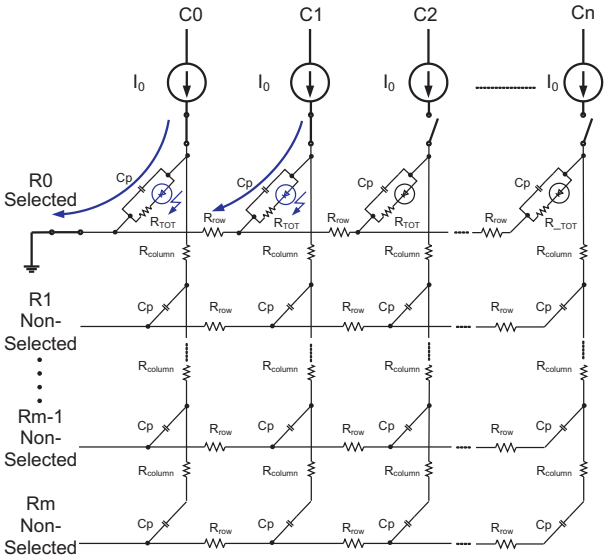

Fig. 2. Schematic electrical circuit of a display

explanation of this phenomenon can be found in [1] and [6]. Here we restrict ourselves to an informal description for self-containment.

A (passive matrix) OLED display has a matrix structure with n rows and m columns. At any crossover between a row and a column there is a vertical diode which works as a pixel. The image itself is given as an integral non-negative $n \times m$ matrix $(r_{ij}) \in \{0, \ldots, \varrho\}^{n \times m}$ representing its RGB values. Consider the contacts for the rows and columns as switches. For the time the switch of row i and column j is closed, an electrical current flows through the diode of pixel (i, j) and it shines. Hence, we can control the intensity of a pixel by the two quantities *electrical current* and *time*. In our application, the electrical current is equal for all pixels. Since high amplitudes of the electrical current or high peaks of intensity respectively, are the major issues with respect to the lifetime of the diodes [5], we try to trade as much time as possible for it. But since an image has to be displayed within a certain time frame T_f, it is a limited resource that we shall use as efficient as possible. Hence, the value r_{ij} determines the amount of time within the time frame in which the switches i and j have to be simultaneously closed. At a sufficient high frame rate e.g. 50 Hz, the perception by the eye is the average value of the light emitted by the pixel and one sees the image.

The traditional addressing scheme is row-by-row. This means that the switch for the first row is closed for a certain time while the switches for the columns are closed for the necessary amount of time dictated by the entries r_{1j}, $j = 1, \ldots, m$. Consequently the first row can be displayed in time $\max\{r_{1j} : j = 1, \ldots, m\}$. Then the second row is displayed and so on.

Consider the schematic image on the left of Fig. 3. Let us compute the amount of time which is necessary to display the image with this addressing scheme. The maxi-

mum value of the entries in the first row is 238. This is the amount of time which is necessary to display the first row. After that the second row is displayed in time 237. In total the time which is required to display the image is $238 + 237 + 234 + 232 + 229 = 1170$ time units.

Now consider the decomposition of the image as the sum of the three images on the right of Fig. 3. In the first image, each odd row is equal to its even successor. This means that we can close the switches for rows 1 and 2 simultaneously, and these two equal rows are displayed in 82 time units. Rows 3 and 4 can also be displayed simultaneously which shows that the first image on the right can be displayed in $82+41$ time units. The second image on the right can be displayed in $155 + 191$ time units while the third image has to be displayed traditionally. In total all three images, and thus the original image on the left via this decomposition, can be displayed in $82+41+155+191+156+38+38 = 701$ time units. This means that we could reduce the necessary time via this decomposition by roughly 40%. We could equally display the image in the original 1170 time units but reduce the peak intensity, or equally the maximum electrical current through a diode by roughly 40%.

$$\begin{pmatrix} 109 & 238 & 28 \\ 112 & 237 & 28 \\ 150 & 234 & 25 \\ 189 & 232 & 22 \\ 227 & 229 & 19 \end{pmatrix} = \begin{pmatrix} 0 & 82 & 25 \\ 0 & 82 & 25 \\ 0 & 41 & 22 \\ 0 & 41 & 22 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 112 & 155 & 3 \\ 112 & 155 & 3 \\ 189 & 191 & 0 \\ 189 & 191 & 0 \end{pmatrix} + \begin{pmatrix} 109 & 156 & 3 \\ 0 & 0 & 0 \\ 38 & 38 & 0 \\ 0 & 0 & 0 \\ 38 & 38 & 19 \end{pmatrix}$$

Fig. 3. An example decomposition

On real-world images, an optimal decomposition of the image allows a reduction of the electrical current to 56% on the average. This means an increase of lifetime by roughly 100%, see [5].

To benefit from this decomposition in practice, an algorithm to solve the optimization problem, which is formally described in Section 2, has to be implemented on a chip which is attached to the display, see Fig. 1. The following design criteria lie in the focus when engineering such an algorithm.

- The algorithm has to react in realtime.
- It must have low hardware complexity allowing small production costs.
- Consequently it has to rely only on a small amount of memory and it should be *fully combinatorial*, i.e. only additions, subtractions, and comparisons are used.

Especially the last of the above criteria clearly establishes a border between our approach and another technique [4] based on *Non-negative Matrix Factorization* [3, 2].

Contributions of this paper

First we show that monochrome images can be optimally decomposed in polynomial time. The presented algorithm has quadratic running time in the worst case. Therefore

we introduce an online version of this algorithm which takes a decision for one row, based on a lookahead of a certain fixed number of rows. This algorithm runs in linear time and has tight competitive ratio. On real world images it turns out that a lookahead of 3 rows gives the most satisfactory results when balancing approximation ratio, ease of implementation and running time. Our computational results show that this algorithm with a lookahead of 3 outperforms the previously best algorithm presented in [1] w.r.t. its practical approximation ratio and even more so w.r.t. its running time. This implies that nearly optimal Doubleline Addressing for real world images can be efficiently computed and, in particular, that an economic hardware implementation meeting the design criteria is possible.

2 The Formal Model

In this section, we will briefly review the formal model presented in [1]. Let $R = (r_{ij}) \in \{0, \ldots, \varrho\}^{n \times m}$ be the matrix representing the image. To decompose R we need to find matrices $F^{(1)} = (f_{ij}^{(1)})$ and $F^{(2)} = (f_{ij}^{(2)})$ where $F^{(1)}$ represents the singleline part and $F^{(2)}$ the two doubleline parts. More precisely, the i-th row of matrix $F^{(2)}$ represents the doubleline covering rows i and $i+1$. Since the overlay (addition) of the subframes must be equal to the original image to get a valid decomposition of R, the matrices $F^{(1)}$ and $F^{(2)}$ must fulfill the constraint $f_{ij}^{(1)} + f_{i-1,j}^{(2)} + f_{ij}^{(2)} = r_{ij}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, m$, where we now and in the following use the convention to simply omit terms with indices running out of bounds. Since we cannot produce "negative" light we require also non-negativity of the variables $f_{ij}^{(\alpha)} \geq 0$. The goal is to find an integral decomposition that minimizes

$$\sum_{i=1}^{n} \max\{f_{ij}^{(1)} : 1 \leq j \leq m\} + \sum_{i=1}^{n-1} \max\{f_{ij}^{(2)} : 1 \leq j \leq m\} \ .$$

This problem can be formulated as an integer linear program by replacing the objective by $\sum_{i=1}^{n} u_i^{(1)} + \sum_{i=1}^{n-1} u_i^{(2)}$ and by adding the constraints $f_{ij}^{(\alpha)} \leq u_i^{(\alpha)}$. This yields

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n} u_i^{(1)} + \sum_{i=1}^{n-1} u_i^{(2)} & & \\
\text{s.t.} \quad & f_{ij}^{(1)} + f_{i-1,j}^{(2)} + f_{ij}^{(2)} = r_{ij} & \text{for all } i,j & \quad (1) \\
& f_{ij}^{(\alpha)} \leq u_i^{(\alpha)} & \text{for all } i,j,\alpha & \quad (2) \\
& f_{ij}^{(\alpha)} \in \mathbb{Z}_{\geq 0} & \text{for all } i,j,\alpha & \quad (3)
\end{aligned}$$

Note that the objective does not contain the f-variables.

Consider the constraints (1) for a fixed column j. By appending the constraint $0 = 0$ and by subtracting the $i-1$-st constraint from the i-th constraint, we obtain the following set of constraints

$$f_{ij}^{(1)} - f_{i-1,j}^{(1)} + f_{ij}^{(2)} - f_{i-2,j}^{(2)} = r_{ij} - r_{i-1,j} \qquad \text{for all } i,j. \quad (4)$$

For each j the constraint-matrix is thus the node-arc incidence matrix corresponding to a graph like in Fig. 4. In the following we refer to this graph, which is solely determined by the number n of rows in the image, by the name *prototype displaygraph* $G_n = (V, A)$.

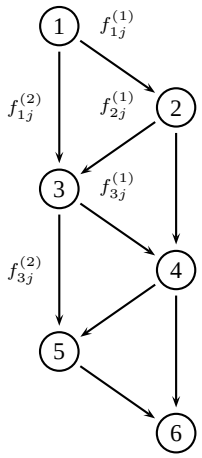

Fig. 4. Prototype displaygraph with variable-names of arcs entering and leaving row 3.

The variables $f_{ij}^{(1)}$ correspond to the arcs going from left to right and vice versa. We call them arcs of type 1. The variables $f_{ij}^{(2)}$ are represented by the vertical arcs, called type 2. The number $r_{ij} - r_{i-1,j}$ is the *demand* $d_j(i)$ of vertex i in column j. The optimization problem can now be understood as follows.

> Given an integer matrix $R \in \mathbb{N}_0^{n \times m}$ reserve capacities $u : A \longrightarrow \mathbb{N}_0$ for the arcs A of G_n such that each of the demands d_j, $j = 1, \ldots, m$ can be *individually* routed in G_n and such that $u(A) = \sum_{e \in A} u(e)$ is minimal.

In this context, *individually* routed means that for any column j the capacities admit a feasible flow satisfying the respective demands d_j.

3 Decomposing Monochrome Images in Polynomial Time

A *monochrome image* is an image $R \in \{0,1\}^{n \times m}$. In this section we show that an optimal decomposition of such an image can be computed in polynomial time. The following example shows the transformation of an image into the demand matrix by the row operations that we described in the previous section.

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \rightsquigarrow \begin{pmatrix} 1 & 0 & 1 \\ -1 & 1 & 0 \\ 1 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \rightsquigarrow \begin{pmatrix} 1 & 0 & 0 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & -1 & -1 \end{pmatrix} \tag{5}$$

Each column is a $0, \pm 1$-vector. Furthermore it is easy to see that the occurrences of 1 and -1 in each column alternate and that each 1 is succeeded by a -1 and each -1 is preceded by a 1 disregarding the zeros inbetween. Moreover, the two matrices on the right of (5) have the same set of feasible solutions with respect to the capacities which are subsets of arcs such that the pairs of nodes $(1, 2)$, $(3, 4)$, $(2, 4)$, and $(1, 4)$ are connected in the corresponding subgraph. Therefore, we assume w.l.o.g. that each column yields exactly one such pair of nodes to which we also refer as a *commodity* in the following. In general the problem of optimally decomposing monochrome images can be understood as follows.

> Given commodities (s_j, t_j), $j = 1, \ldots, m$ with $s_j < t_j$ for each j and a number n, select a minimal number of arcs of G_n such that there exists a path from s_j to t_j for each $j = 1, \ldots, m$.

The nodes s_j are called *sources* and the nodes t_j are called *sinks*. Furthermore, if a node is neither a source nor a sink, we call it *Steiner*. The selection of arcs of G_n is given by a function $u : A \to \{0, 1\}$, where $u(a) = 1$ if the arc a is selected and $u(a) = 0$ otherwise.

The next lemma is easy to prove but crucial to obtain a polynomial-time algorithm. Here $u(\delta^{out}(i))$ denotes the number of selected arcs leaving node i. Similarly, $u(\delta^{in}(i))$ denotes the number of selected arcs entering i.

Lemma 1. *Given a feasible solution u, then there exists a feasible solution u' with*

$$\begin{aligned} u'(\delta^{out}(i)) &\leq 1 \qquad \textit{and} \\ u'(\delta^{in}(i)) &\leq 1 \end{aligned} \tag{6}$$

with the same total weight.

Proof. It is easy to see that we remain feasible if we substitute an arc of type 2 by the two arcs of type 1 incident to head and tail respectively. We do not change the number of selected arcs by selecting the other type 1 arc instead of the type 2 arc if $u(\delta^{out}(i)) > 1$ or $u(\delta^{in}(i)) > 1$ respectively as depicted in Fig. 5. Such a replacement is feasible, since each pair of nodes which was connected by a path before the replacement is still connected after the replacement.

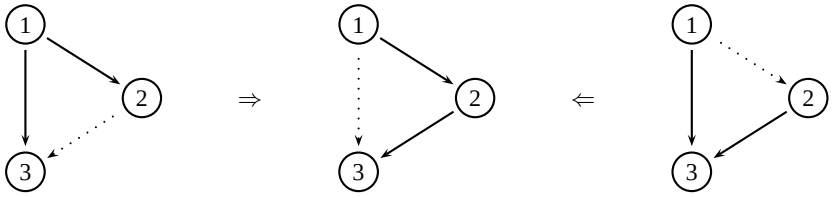

Fig. 5. Transformation to maintain the degree condition

In the forthcomming we maintain $u(\delta^{out}(i)) \leq 1$ and $u(\delta^{in}(i)) \leq 1$ as an invariant and call it *degree condition*. Thereby, the selection of one outgoing arc uniquely transforms the instance to the same problem with one row less. However, if we have selected the outgoing arc of type 2 for node 1 and if node 2 is a source, we have to select the arc of type 2 as well to leave the second node to maintain the degree condition. In turn, this might force the same for the outgoing arcs of node 3 and so on. These implications evolve until either everything up to the current row is balanced or an odd commodity, say (s, t) with $t - s$ odd, produces a conflict (see Fig. 6). More precisely speaking, *balanced up to row* i means that the assignment to the capacities of the arcs a such that $head(a) \leq i$ is a feasible solution to the subinstance consisting of the rows $1, \ldots, i-1$ of the given image. Hence, the solution of the subproblem starting at node i does not depend on how we have balanced up to row i. Note that both subproblems are considered with respect to the given image, i.e. all commodities (s_j, t_j) with $s_j < i < t_j$ are split into (s_j, i) and (i, t_j) where the former commodity is considered with the first subproblem and the latter with the second. It is easy to see that feasible solutions to the subproblems join to feasible solutions for the original problem. In particular, balanced up to row i implies that we may forbid the arc $(i-1, i+1)$ and remain feasible.

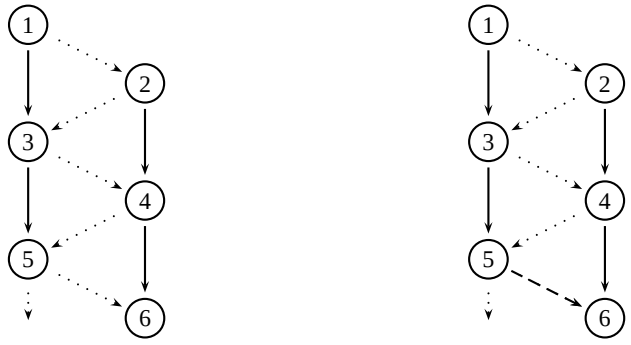

Fig. 6. In both examples, we are given the commodities $(1, 3)$ and $(2, 6)$. On the left, we additionally have $(3, 5)$ whereas on the right it is the odd commodity $(3, 6)$ instead. While the left example is balanced up to node 6, the commodity $(3, 6)$ produces a conflict on the right.

We will now present a basic dynamic programming scheme using node labels to store at node i how much it costs to balance up to node i. The label of the first node is 0 and all others are ∞ at the beginning. Let i be the current node. Assume that it is a source since otherwise we could simply skip it and proceed with the same label at node $i + 1$. We select arcs of type 2 until we either find a conflict or a node, say t, up to which we are balanced. If the label of i plus the number of selected arcs is smaller than the label of t, we update it accordingly. For a later reconstruction of the solution, we also store i as the predecessor of t. In case of a conflict we do not update anything. Afterwards, we proceed to node $i+1$. If the label of $i+1$ is more than 1 greater than label i, we also update it and set i as the predecessor of $i + 1$, i.e. selecting the arc of type 1. Then we repeat these steps until we reach the end. Because of the degree condition, we

can transform any instance such that each node is the source of at most one commodity and also the sink of at most one commodity. Hence, by a preprocessing of the input data which takes $O(n \cdot m)$ time, we can annotate the nodes with the necessary information. Since every step involves the visit of $O(n)$ nodes and arcs, the total time for computing the capacities is $O(n^2)$.

Theorem 1. *The optimal decomposition of an image given by a matrix $R \in \{0,1\}^{n \times m}$ can be computed in $O(n \cdot m + n^2)$.*

4 The Online Problem

Recall that we intend to develop an algorithm that finds a decomposition in realtime while keeping it simplistic enough such that it can be implemented on a chip with a low hardware complexity. Hence, we are looking for a linear time algorithm that uses only additions, subtractions, and comparisons. Since we do not want to scan over the whole rest of the graph in each iteration, it is natural to restrict the lookahead to a certain number of rows. It follows an online version of our problem where we have to fix the capacities of the outgoing arcs of a node only based on the knowledge of the following c rows. Again, we consider monochrome images first. At the end of this section we describe how our method can be adapted to decompose arbitrary colored images.

The canonical algorithm uses the one of Sec. 3 as follows. We solve the instance of the known c rows to optimality. According to that solution, we fix the outgoing arcs of the first node. After updating the instance and reading the next row, we repeat these steps until we reach the end. The computation takes $O(c \cdot n)$ time disregarding the time for the preprocessing that we have to spend anyways to parse the input.

In the following, we will first give a lower bound on the competitive ratio of any algorithm in that online setting. Afterwards, we will analyze the competitive ratio of the aforementioned approach. Before we state the theorem, it is helpful to have a look at following example where the adversary starts with the image in the middle and then reveals the fourth row according to the arc we have selected for the first node.

$$\begin{pmatrix} \not{1} & \not{0} & \not{0} \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \xleftarrow{\text{type 1}} \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ \star & \star & \star \end{pmatrix} \xrightarrow{\text{type 2}} \begin{pmatrix} \not{1} & \not{0} & \not{0} \\ \not{1} & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \tag{7}$$

The optimal value in both cases is 3. But after making the choice for the first row, the adversary force us to pay 4.

Theorem 2. *Any online algorithm that fixes the outgoing arcs of node i without knowing the rows $i + c$ and beyond, has a competitive ratio of at least $\frac{c+1}{c}$.*

Proof. An adversary reveals the first c nodes of an instance with the commodities (i, t_i) forall $i = 1, \ldots, c$ where $t_i \in \{c+1, c+2\}$ is chosen later depending which arc the algorithm picks following the idea shown in (7). If the algorithm selects the arc of type 1, then the adversary sets t_1 to the odd value. Otherwise, it is set to the even node. All other t_i are set such that the commodities (i, t_i) are even. The optimal solution of the

residual problem is c. Hence, the achieved objective value is $c+1$ whereas the opposite choice for the first arc would yield an optimal solution of value c.

Theorem 3. *There is an algorithm with competitive ratio* $\frac{c+1}{c}$.

Proof. We first compare the optimal algorithm running on the complete instance with the one running on the first c rows. Let t be the node such that the selection of the type 2 arcs starting with the first node gets balanced with respect to the whole instance. If no such node exists, then in every feasible solution the arc of type 1 has to be picked for leaving node 1. This also holds for the instance restricted to the first c nodes. Note that the choice of the first arc only depends on the rows strictly less than t. Hence, if $t \leq c+1$, then the online algorithm makes the same decision on the first arc as the optimal one. Otherwise, it takes the arc of type 1. So let us assume that the arc of type 2 would have been the optimal choice. Hence, the optimal label of node t is $t-2$. On the other hand, choosing only arcs of type 1 yields a label of $t-1$. Since $t > c+1$, the ratio $\frac{t-1}{t-2} \leq \frac{c+1}{c}$. Since we can partition the solution that is found by the optimal algorithm into independent balanced parts, we can repeat these arguments on them.

4.1 A compact $4/3$-Approximation

In this subsection, we unroll the generic algorithm for the case $c = 3$ and give a compact set of rules for the selection of the capacities. These rules can be generalized to decomposed colored images yielding a competitive approximation algorithm in practice. They are described as follows and depicted in Fig. 7.

Compact We consider the first three nodes. If the first node is not a source, we skip it without selecting any outgoing arc. Assume it is a source in the following. If the corresponding sink is node 2 (see Fig. 7a), we select the arc of type 1. If node 2 is a *Steiner node* (Fig. 7b), i.e. it is neither a source nor a sink, then we select the arc of type 2. If node 2 is a source and node 3 is either the corresponding sink or Steiner (Fig. 7c/d), we select the arc of type 1. Otherwise, we select the arc of type 2 (Fig. 7e).

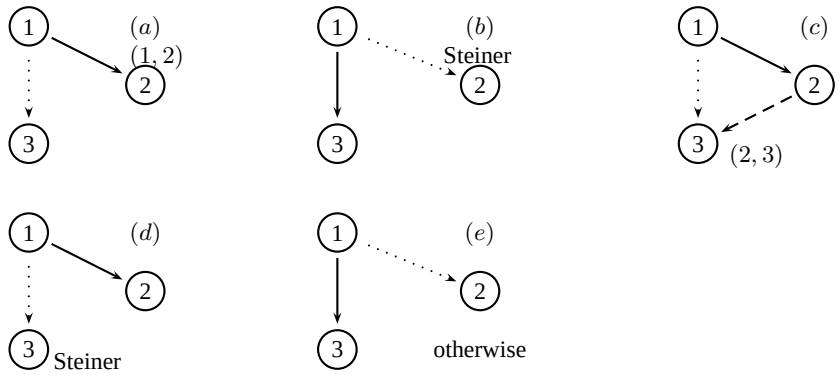

Fig. 7. The rules for the algorithm COMPACT

Lemma 2. *The algorithm* COMPACT *achieves a competitive ratio of* $4/3$.

Proof. The interesting cases are if node 1 and 2 are sources and their corresponding sinks are not revealed yet. Assume first that node 3 is Steiner. We show that we can transform every feasible solution such that the Steiner node is isolated (see Fig. 8). Consider a feasible solution where the arc between node 1 and 3 is picked. Since node 2 is a source the arc between 2 and 4 is also selected. Moreover there must be an arc between node 3 and 5. We can reconnect the tail of latter to node 4 and the head of the outgoing arc of node 1 to node 2. Thereby, we do not change the number of arcs and the routing remains feasible. If node 3 is a source instead, the demand of node 1 may go piggyback with the demand of node 2 or with the one of node 3. Since the first three nodes are sources, each of them has an outgoing arc in every feasible solution. If an adversary reveals that our decision to take the arc of type 2 for node 1 was wrong, we need one surplus arc to fix it. Until the adversary does not force us to change the parity, i.e. choose an arc of type 1, we do not use more arcs than optimal. Moreover, if we are forced to take such an arc, the problem decomposes into independent subproblems. Thereby, we use at most one surplus arc by three necessary ones and hence get a ratio of $4/3$.

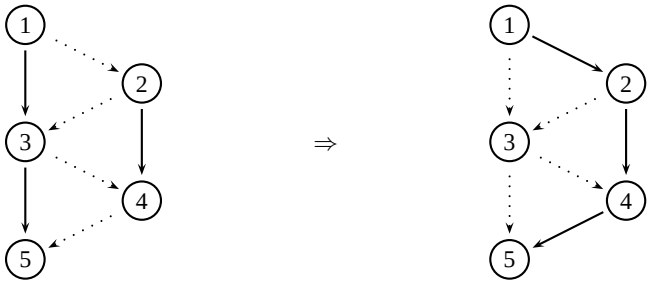

Fig. 8. Isolating a Steiner node

Generalizing to colored images Recall that in the general case the instance is not given by a binary matrix but as $R = (r_{ij}) \in \{0, \ldots, \varrho\}^{n \times m}$. So we need to generalize our concepts for this purpose. We briefly sketch how this is done in our algorithm. Whenever $\max\{r_{ij} - r_{i-1,j} : 1 \leq j \leq m\} > 0$ we call the node i a source in the prototype displaygraph. For the ease of notation, we use the following abbreviation $\overline{r_i - r_{i-1}}$ for the maximum over all columns. Similarly, we call node i a sink whenever $\overline{r_{i-1} - r_i} > 0$. The degree condition transforms into $u(\delta^{out}(i)) \leq \overline{r_i}$ and $u(\delta^{in}(i)) \leq \overline{r_{i-1}}$. Similarly to the set of rules presented above, we define five rules for the general case. The rule a) for example translates into the rule in which we have to reserve a capacity of at least $\overline{r_2 - r_1}$ on the arc of type 1 leaving node 1. The other rules can be generalized accordingly. We do not know the exact approximation ratio of this generalized algorithm. In particular we do not know whether it exceeds $4/3$. However, as the computational results of the next section show, it behaves very well in practice.

5 Computational Results

We use the same testset and machine as in [1]. It is a Pentium M with 2GHz and 2MB L2 cache. The images are portraits of 197 employees of the MPI. They have a resolution of 180×240 pixels and a colordepth of 24 bit, i.e. $n = 180$, $m = 720$, and $\varrho = 255$. From [1], we take the algorithms called *ec-bf-mcgu-2* and *ec-bf-mcgu-4* which performed best there. They differ only by the fact that the former combines two rows and the latter combines up to four rows. We compare them to the generalized COMPACT algorithm (that solves the doubleline problem) and a cascading of it such that four or two rows may be combined. We will elaborate on the differences of *ec-bf-mcgu-4* and CASCADING to the doubleline addressing scheme at the end of this section.

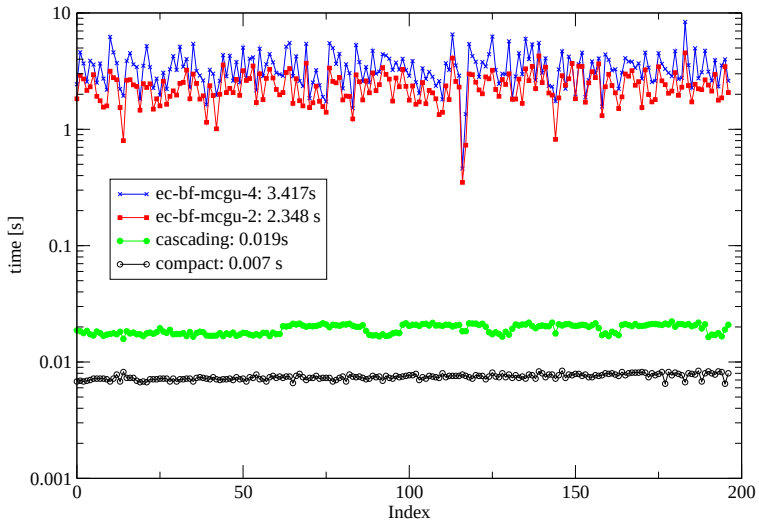

Fig. 9. Running time comparison of the old and new algorithms

In Fig. 9, we show the running times for each instance. The squares and the crosses (top) represent the old measurements of *ec-bf-mcgu-2* and *ec-bf-mcgu-4* respectively. Whereas the dots and circles (bottom) correspond to the new algorithms COMPACT and CASCADING. We connected the measurements by lines to guide the eye. Note the logarithmic scale of the time axis. One can see that the new algorithms are two orders of magnitudes faster than the old ones. Comparing the mean running times, the improvement is more than a factor of 300 between *ec-bf-mcgu-2* and COMPACT, and about 180 between *ec-bf-mcgu-4* and CASCADING. Moreover, the variance decreases drastically. This is due to the fact that the running time of the old algorithms depends strongly on the input data, i.e. on the unary encoding length, while it scales only with the size of the binary encoding length in the new ones.

It remains to show that the drastic improvements with respect to the running times are not at the cost of the approximation quality. Therefore, we solved the corresponding

integer linear programs for doubleline addressing with the commercial solver CPLEX. Thereby, we obtained the optimal solutions and were able to compare the per-instance approximation ratios of *ec-bf-mcgu-2* and COMPACT. The results are depicted in Fig 10.

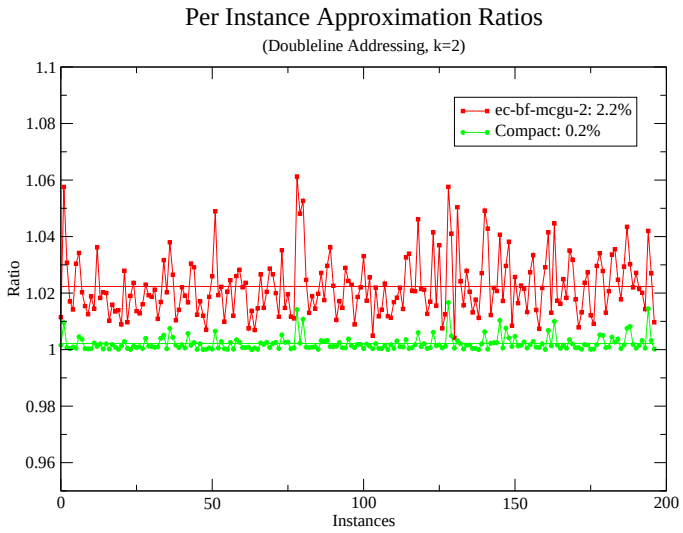

Fig. 10. Results of the old and the new doubleline algorithms normalized to the corresponding optimal solution. The horizontal lines indicate the respective means.

As one can see, the quality of the approximation of the new algorithm is not worse than for the one presented in [1]. In fact, on average it is even considerably better. Recall that the objective of our optimization problem is proportional to the electrical current and therefore has a direct impact on the lifetime of such a passive matrix OLED display. The average gap of 0.2% shows that we have found an algorithm that does not leave much room for improvement with respect to the necessary electrical current to display real world images using the doubleline addressing scheme.

However, one can consider combining more than two rows to reduce the electrical current even further. The heuristics of [1] have been implemented in such a more general way, that the number of lines up to which we want to combine them, is controlled by a parameter $k = 2, 3, 4, \ldots$ whereas COMPACT is specialized to the doubleline addressing scheme, i.e. $k = 2$. Nevertheless, doubleline addressing is an important building block for more advanced strategies. We outline here a simple one that is achieved by cascading COMPACT. This means that we take the two frames that contain the computed doublelines and feed both independently as input to COMPACT again. Thereby two doublelines of the outcome of the first phase may potentially be combined to a doubleline which represents the combination of four lines with respect to the original image. Thereby, we push forward into the range of *ec-bf-mcgu-4* without considering

the combination of three lines. The ratios of the objectives of CASCADING and *ec-bf-mcgu-4* for each instance are presented in Fig. 11.

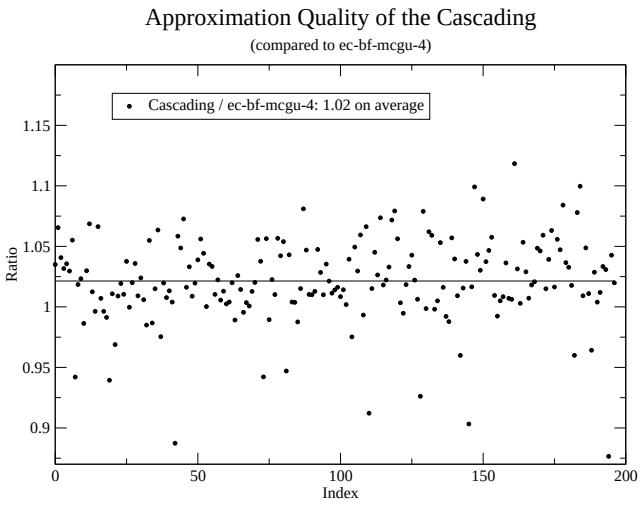

Fig. 11. The relative objective of CASCADING with respect to *ec-bf-mcgu-4*. The horizontal line indicates the average ratio of 1.02

The slightly worse average approximation ratio by a factor of 1.02 is more than compensated by the improvement with respect to the running time by a factor of about 180. Moreover, it is not possible that CASCADING yields a worse objective value than COMPACT on the same instance whereas this behavior occurred on some instances concerning *ec-bf-mcgu-2* and *ec-bf-mcgu-4*. However, there might be other strategies that are simplistic enough to guarantee a fast running time at low hardware complexity to close the gap. This is subject to ongoing research.

We want to conclude with a brief discussion of the applicability of consecutive Multiline Addressing to a broader set of images. Based on the results for human faces, it is natural to ask for photographs in general. It turned out that on a variety of over 3000 pictures, COMPACT achieves a reduction of the electrical current to 56% on the average with a mean per instance approximation ratio of 1.003 compared to the optimal solution provided by CPLEX. The results for two exemplary music videos are even better with a reduction to 51% which is only a factor of 1.002 away from the optimum. We explain this behavior by the fact that the content of photos is rather smooth, e.g. they are not dominated by sharp edges as in artificial images like cliparts and text by bitmap fonts. This is in agreement with the results on a testset of wallpapers for mobile phones with a mean reduction to 63% and approximation factor of 1.005 averaged over about 4500 samples. We observed that diagonal lines, in particular if the width is only one pixel and the contrast to the neighborhood is high, constitute an obstacle to Multiline Addressing.

Finally, we want to thank Markus Tetzlaff for providing us with the large set of his digital photos and Tobias Jung for performing the tests as part of his bachelor thesis.

References

1. Friedrich Eisenbrand, Andreas Karrenbauer, Martin Skutella, and Chihao Xu. Multiline Addressing by Network Flow. In Thomas Erlebach and Yossi Azar, editors, *Proceedings of the 14th Annual European Symposium on Algorithms (ESA 06)*, volume 4168 of *Lecture Notes in Computer Science*, pages 744–755, Zürich, Switzerland, 2006. Springer.
2. D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, 13, 2001.
3. P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111–126, 1994.
4. E. Smith, P. Routley, and C. Foden. Processing digital data using non-negative matrix factorization. *patent GB 2421604A, pending*, 2005.
5. K. M. Soh, C. Xu, and C. Hitzelberger. Dependence of OLED Display Degradation on Driving Conditions. *Proceedings of SID Mid Europe Chapter Fall Meeting*, 2006.
6. C. Xu, A. Karrenbauer, K. M. Soh, and J. Wahl. A New Addressing Scheme for PM OLED Display. *Society for Information Display (SID) Symposium Digest*, 2007.
7. C. Xu, J. Wahl, F. Eisenbrand, A. Karrenbauer, K. M. Soh, and C. Hitzelberger. Verfahren zur Ansteuerung von Matrixanzeigen. *Patent 10 2005 063 159, Germany, pending*, 2005.