RLE-based Algorithm for Testing Biorders

Oliver Lanzerath

Department of Computer Sciences, Bonn-Rhein-Sieg University of Applied Sciences,
Grantham-Allee 20, 53757 Sankt Augustin, Germany
oliver.lanzerath@smail.inf.hochschule-bonn-rhein-sieg.de
http://www.inf.h-brs.de

Abstract. Binary relations with certain properties such as biorders, equivalences or difunctional relations can be represented as particular matrices. In order for these properties to be identified usually a rearrangement of rows and columns is required in order to reshape it into a recognisable normal form. Most algorithms performing these transformations are working on binary matrix representations of the underlying relations. This paper presents an approach to use the RLE-compressed matrix representation as a data structure for storing relations to test whether they are biorders in a hopefully more efficient way.

Keywords: RLE-XOR · RLE-permutation · biorder

1 Introduction

The matrix representation of a binary relation can be interpreted as a bitmap image, that is, a bit sequence. In many cases the usage of a *run length encoding* (RLE) technique results in a smaller representation of such pictures by shorter codes for lengthy bit strings. Hence, algorithms which use RLE-compressed binary matrices as input may have better runtime complexity on average. A bitvector $\boldsymbol{x}$ consists of alternating series of **0**- and **1**-sequences. Referring to [3], a bitvector can be represented by the lengths of the single sequences as follows.

Run Length Encoding. Let $seq_i \in \left\{\mathbf{0}^j | j \in \mathbb{N}\right\} \cup \left\{\mathbf{1}^j | j \in \mathbb{N}\right\}$, $i \in \mathbb{N}$, be a sequence with $value(seq_i) = 0, seq_i \in \left\{\mathbf{0}^j | j \in \mathbb{N}\right\}$ and $value(seq_i) = 1, seq_i \in \left\{\mathbf{1}^j | j \in \mathbb{N}\right\}$. Then, a bitvector $\boldsymbol{x} = x_0 ... x_{n-1} \in \{0,1\}^n$ can be represented as $\boldsymbol{x} = seq_1 ... seq_k$, $1 \leq k \leq n, value(seq_i) \neq value(seq_{i+1}), \sum_{i=1}^{k} |seq_i| = n$. The RLE-compression of a vector $\boldsymbol{x}$ is given by the vector

$$\boldsymbol{x}^{rle} = x_0 \left[|seq_1|, ..., |seq_k|\right] \tag{1}$$

Figure 1 shows an example for the RLE-compression of a bitvector. Furthermore we make use of a notation similar to arrays for accessing elements of the RLE-compressed vector, where $\boldsymbol{x}^{rle}[0]$ refers to the leading element x_0 and $\boldsymbol{x}^{rle}[i], 1 \leq i \leq k$, to the following length specifications.[1]

[1] E.g. the RLE-compression of the vector $\boldsymbol{x} = 1100001$ is given by $\boldsymbol{x}^{rle} = 1\,[2,4,1]$ with $\boldsymbol{x}^{rle}[0] = 1, \boldsymbol{x}^{rle}[1] = 2, \boldsymbol{x}^{rle}[2] = 4$ and $\boldsymbol{x}^{rle}[3] = 1$.

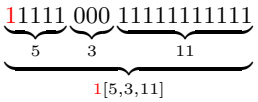

Fig. 1. Example for the RLE-compression of a bitvector.

Biorders. *Biorders* can be defined in different ways. First we introduce the formal definition[4]: Let $R \subseteq X \times X$ be a homogeneous binary relation. R is called biorder (or: Ferrers relation in heterogeneous cases), iff

$$aRb \wedge cRd \wedge \neg aRd \rightarrow cRb$$

holds $\forall\, a, b, c, d \in X$.

In the context of this paper the following definiton is helpful. A binary relation is called a biorder, iff the matrix can be represented in (upper left) *echelon block form*[2] by rearranging rows and columns independently [4]. Thus, it is sufficient to test relations for being biorders by using an algorithm that computes the echelon block form if possible and returns an error otherwise. Figure 2 shows the results of the group stage of group B during the recent FIFA World Cup, where ARB, iff A won the match against B, $A, B \in \{ESP, CHL, NLD, AUS\}$, $A \neq B$. By rearranging rows and then columns the matrix for the relation R

R	*ESP*	*CHL*	*NLD*	*AUS*
ESP	0	0	0	1
CHL	1	0	0	1
NLD	1	1	0	1
AUS	0	0	0	0

R	*ESP*	*CHL*	*NLD*	*AUS*
NLD	1	1	0	1
CHL	1	0	0	1
ESP	0	0	0	1
AUS	0	0	0	0

R	*AUS*	*ESP*	*CHL*	*NLD*
NLD	1	1	1	0
CHL	1	1	0	0
ESP	1	0	0	0
AUS	0	0	0	0

Fig. 2. Group stage FIFA World Cup 2014 (group B).

is transformed into echelon block form proving that R is a biorder.[3] If a given relation is a biorder, the echelon block form can be achieved in two steps [1]:

1. Sort the rows by their *Hamming weight*[4] in descending order.
2. Sort the columns by their Hamming weight in descending order.

A corresponding algorithm would perform these two steps and check if the result is in echelon block form. The time complexity of such an algorithm is in $\mathcal{O}(2n \log n + n^2) \subseteq \mathcal{O}(n^2)$. The following lemma states, that with respect to biorder tests the second step is not needed.

[2] Visually speaking, a binary matrix is in upper left echelon block form, if all 1-entries are placed in the upper left corner and the cardinality of the 1-entries monotonically decreases from top to bottom.

[3] Soccer results do not induce biorders in general. The example is well-chosen here.

[4] We use the notation $\|\boldsymbol{x}\| = \sum_{i=0}^{n-1} x_i$ to note the Hamming weight of a bitvector $\boldsymbol{x}$ (cf. [2]).

Lemma 1. *Let $\boldsymbol{A}$ be a binary $n \times n$-matrix, and let $\boldsymbol{A'}$ be the result of sorting the rows of $\boldsymbol{A}$ by their Hamming weights in descending order. $\boldsymbol{A}$ is a biorder if and only if $\boldsymbol{A'}$ is exclusively composed of column vectors $\boldsymbol{x}_{*i} \in \{0^n, 1^n\} \cup \left\{1^j 0^{n-j} | 1 \leq j \leq n\right\}$.*

Proof. First, we show the if-part by contradiction. Assume that a given $n \times n$-matrix resp. binary relation is a biorder, and after having sorted the rows by their Hamming weights there exists a column vector that does not fulfil the condition. Then there exists at least one column vector in which a **1**-sequence follows a **0**-sequence, i.e. there exists at least one i, $1 \leq i \leq n$, such that $\boldsymbol{x}_{*i} \in \left\{\upsilon 01\omega | \upsilon \in \{0,1\}^j, \omega \in \{0,1\}^{n-j-2}, 0 \leq j \leq n-2\right\}$. Now we sort the columns by their Hamming weights. Let $x_{ki} = 0$ and $x_{li} = 1$ with $k < l$, then $\|\boldsymbol{x}_{k*}\| \geq \|\boldsymbol{x}_{l*}\|$. After having sorted rows and columns the matrix should be in echelon block form because it is a biorder. Then $x_{l0} = x_{l1} = \cdots = x_{li} = 1$ and $x_{k0} = x_{k1} = \cdots = x_{ki} = 1$ because of $\|\boldsymbol{x}_{k*}\| \geq \|\boldsymbol{x}_{l*}\|$. This contradicts the assumption.

The other direction is obvious. If the rows are ordered by Hamming weight and all column vectors are of the desired type, there must be an ordering $\langle \boldsymbol{x}_{*j_1}, \boldsymbol{x}_{*j_2}, ..., \boldsymbol{x}_{*j_n} \rangle$, $j_i \in \{0, ..., n-1\}$ of the column vectors with $\|\boldsymbol{x}_{*j_1}\| \geq \|\boldsymbol{x}_{*j_2}\| \geq \ ... \ \geq \|\boldsymbol{x}_{*j_n}\|$. The echelon block form is accomplished by sorting the columns according to this ordering.

2 Algorithm

The complexity of checking all column vectors of a binary matrix for a certain form is still in $\mathcal{O}(n^2)$. If the column vectors are RLE-compressed, the checking can be done in linear time. We only need to verify that all RLE-compressed column vectors have a length of 1 or 2 and start with a **1**-sequence. We assume the relation is represented in RLE-compressed form, i.e. column vectors as well as row vectors are RLE-compressed (c.f. running example in Fig. 3). A biorder-

	ESP	*CHL*	*NLD*	*AUS*
ESP	0	0	0	1
CHL	1	0	0	1
NLD	1	1	0	1
AUS	0	0	0	0

ESP	0[3,1]
CHL	1[1,2,1]
NLD	1[2,1,1]
AUS	0[4]

ESP	*CHL*	*NLD*	*AUS*
0[1,2,1]	0[2,1,1]	0[4]	1[3,1]

Fig. 3. RLE-compressed rows and columns

checking algorithm is defined as follows. First one adjusts the definition of the Hamming weight for RLE-compressed vectors with respect to the new sorting algorithm. If the leading element $\boldsymbol{x}^{rle}[0]$ is 1, all odd positions in the following array refer to 1-entries in the corresponding binary matrix, and vice versa.

Hence, the Hamming weight for RLE-compressed vectors can be defined as:[5]

$$\|\boldsymbol{x}^{rle}\| = \begin{cases} \sum\limits_{i=1, i\in\mathbb{O}_+}^{|\boldsymbol{x}^{rle}|} \boldsymbol{x}^{rle}[i], & \boldsymbol{x}^{rle}[0] = 1 \\ \sum\limits_{i=2, i\in\mathbb{E}_+}^{|\boldsymbol{x}^{rle}|} \boldsymbol{x}^{rle}[i], & \boldsymbol{x}^{rle}[0] = 0 \end{cases} \tag{2}$$

where $\mathbb{O}_+$ and $\mathbb{E}_+$ are the positive odd and even numbers, respectively.

Sorting RLE-compressed row vectors by the Hamming weight causes changes in the column vectors, too. In binary-coded cases this is no point of interest because this change occurs automatically. But, if the vectors are RLE-compressed, it is much more complicated. A permutation of the row vectors $\boldsymbol{x}^{rle}_{j*}$ can have effects on all column vectors $\boldsymbol{x}^{rle}_{*i}$. Assume $\boldsymbol{x}_{j*}$ and $\boldsymbol{x}_{k*}$ are swapped then x_{ji} and x_{ki} must be inverted iff $x_{ji} \neq x_{ki}$. Algorithm 1 implements a bit-compare function for RLE-compressed vectors.

Data: A RLE-compressed n-bitvector $\boldsymbol{x}^{rle}$ and two integers $j, k, 1 \leq j \leq n, 1 \leq k \leq n, j < k$
Result: Boolean: $x_k \neq x_j$
$dist := 0$;
$counter := \boldsymbol{x}^{rle}[1]$;
for $i := 2$ **to** $|\boldsymbol{x}^{rle}|$ **do**
  if $counter \geq j \ \&\& \ counter \leq k$ **then**
    $dist++$;
  end
  $counter+ = \boldsymbol{x}^{rle}[i]$;
end
if $dist == 1 \ mod \ 2$ **then**
  return $true$;
else
  return $false$;
end

Algorithm 1: $bitCompare$-function for RLE-compressed n-bitvectors

To ensure that in case $x_{ji} \neq x_{ki}$ the corresponding bits must be switched, we use an XOR-operation with a special bit mask:

$\boldsymbol{x}$	x_1	$\cdots$	x_{j-1}	x_j	x_{j+1}	$\cdots$	x_{k-1}	x_k	x_{k+1}	$\cdots$	x_n
mask	0	$\cdots$	0	1	0	$\cdots$	0	1	0	$\cdots$	0
XOR	x_1	$\cdots$	x_{j-1}	$\overline{x_j}$	x_{j+1}	$\cdots$	x_{k-1}	$\overline{x_k}$	x_{k+1}	$\cdots$	x_n

The logical XOR is the essential part of the whole procedure. Algorithm 2 shows how the XOR-operation can be computed on RLE-compressed vectors.

[5] $|\boldsymbol{x}^{rle}|$ is the length of the array or the RLE-compressed vector, respectively.

First the biorder checking algorithm sorts the row vectors $\boldsymbol{x}_{j*}^{rle}$ by their Hamming weights. For each pairwise permutation of rows all column vectors $\boldsymbol{x}_{*i}^{rle}$ must be adjusted if necessary. After that the resulting column vectors must be checked for meeting the conditions of Lemma 1. If each column vector meets the conditions, the underlying relation is a biorder.

3 Examination

As is well known, sorting row vectors of a binary matrix by the Hamming weight is in $\mathcal{O}(n \log n)$, as well as sorting RLE-compressed vectors. But, to check the conditions of Lemma 1 in later process, the column vectors need to be adjusted, too. In the worst case all n column vectors must be modified for each change of rows and, therefore, n XOR-operations have to be computed. The required time for each XOR-operation depends on the length of the RLE-compressed vectors. Hence, the time complexity of the sorting procedure is bounded from above by $\mathcal{O}(n^3 \log n)$. Now, the test for meeting the conditions of Lemma 1 can be done in $\mathcal{O}(n)$.

Data: Two RLE-compressed copies of n-bitvectors $\boldsymbol{x}^{rle},\ \boldsymbol{y}^{rle}$
Result: Result of the XOR $\boldsymbol{z}^{rle}$
$x_{pointer} := 1;$
$y_{pointer} := 1;$
$z_{pointer} := 1;$
$\boldsymbol{z}^{rle}[0] := |\boldsymbol{x}^{rle}[0] - \boldsymbol{y}^{rle}[0]|;$
while $\sum_{i:=1}^{z_{pointer}} \boldsymbol{z}^{rle}[i] < n$ **do**
  if $\boldsymbol{x}^{rle}[x_{pointer}] == \boldsymbol{y}^{rle}[y_{pointer}]$ **then**
    $\boldsymbol{z}^{rle}[z_{pointer}] += \boldsymbol{x}^{rle}[x_{pointer}];$
    $x_{pointer}++;$
    $y_{pointer}++;$
  else if $\boldsymbol{x}^{rle}[x_{pointer}] < \boldsymbol{y}^{rle}[y_{pointer}]$ **then**
    $\boldsymbol{z}^{rle}[z_{pointer}] += \boldsymbol{x}^{rle}[x_{pointer}];$
    $\boldsymbol{y}^{rle}[y_{pointer}] -= \boldsymbol{x}^{rle}[x_{pointer}];$
    $x_{pointer}++;$
    $z_{pointer}++;$
  else
    $\boldsymbol{z}^{rle}[z_{pointer}] += \boldsymbol{y}^{rle}[y_{pointer}];$
    $\boldsymbol{x}^{rle}[x_{pointer}] -= \boldsymbol{y}^{rle}[y_{pointer}];$
    $y_{pointer}++;$
    $z_{pointer}++;$
end

Algorithm 2: XOR on RLE-compressed n-bitvectors

To speed up the runtime of the algorithm one could use a more specific operation than the XOR. Furthermore, generating customized bitmasks for each column during the sorting process could reduce the number of necessary XOR-operations to n. In this paper the logical XOR on RLE-compressed vectors is focused because using it in context of checking relations for being biorders is only one possible use case. An other one could be a simple compare function that returns $true$ if two RLE-compressed vectors are equal and $false$ otherwise. Premising two n-bitvectors $\boldsymbol{x}^{rle}, \boldsymbol{y}^{rle}$ are equal, the bitwise XOR requires n steps and the RLE-XOR requires $|\boldsymbol{x}^{rle}| \cdot 32$ steps (assuming that natural numbers are saved as 32-bit integers). Hence, such an algorithm can reach much better runtimes for sparsely populated or sorted vectors than a bitwise one. A conceivable use case for such an algorithm could be an automated error correction for particular kinds of databases.

4 Conclusion

Advantages and disadvantages of using RLE codes in the context of relational algebraic methods have not been well studied yet. The main contribution of this paper is to motivate further research and analysis of RLE encodings for binary relations. Actually, the presented algorithm is in $\mathcal{O}(n^3)$ but clearly conveys the advantages of using RLE-codes. Our current work focuses on the development of further efficient algorithms for logical operators on RLE codes. Once a sufficient repository is available, these operations can be used for more efficient row-to-row and column-to-column comparison and, hence, sorting. Together with more test procedures for difunctionality and transitivity, we hope to find an algorithm that comes close to the "magic" runtime complexity of $\mathcal{O}(n^{2.3})$.

Acknowledgements. The author wish to express his thanks to Dr. Martin E. Müller and Professor Dr. Kurt-Ulrich Witt for careful reading and useful comments.

References

1. Müller, M.E.: Towards Finding Maximal Subrelations with Desired Properties. In: Höfner, P., Jipsen, P., Kahl, W., Müller, M.E. (eds.) RAMiCS 2014. LNCS, vol. 8428, pp. 344–361. Springer, Heidelberg (2014)
2. Reed, I.: A class of multiple-error-correcting codes and the decoding scheme. Transactions of the IRE Professional Group on Information Theory, vol. 4, no. 4, pp. 38–49 (1954)
3. Salomon, D.: Data compression - The Complete Reference, 4th Edition. Springer, London (2007)
4. Schmidt, G.: Relational Mathematics, Encyclopedia of Mathematics and its Applications, vol. 132. Cambridge University Press, Cambridge (2011)