



## Optimal conclusive sets for comparator networks

Guy Even<sup>a</sup>, Tamir Levi<sup>b,\*</sup>, Ami Litman<sup>b</sup>

<sup>a</sup> School of Electrical Engineering, Tel-Aviv University, Tel-Aviv 69978, Israel

<sup>b</sup> Faculty of Computer Science, Technion, Haifa 32000, Israel

### ARTICLE INFO

Dedicated to the memory of Professor Shimon Even

#### Keywords:

Zero-one Principle  
Comparator networks  
Sorting networks  
Bitonic sorting  
Merging networks

### ABSTRACT

A set of input vectors  $S$  is conclusive for a certain functionality if, for every comparator network, correct functionality for all input vectors is implied by correct functionality for all vectors in  $S$ . We consider four functionalities of comparator networks: sorting, merging, sorting of bitonic vectors, and halving. For each of these functionalities, we present two conclusive sets of minimal cardinality. The members of the first set are restricted to be binary, while the members of the second set are unrestricted. For all the above functionalities, except halving, the unrestricted conclusive set is much smaller than the binary one.

© 2009 Elsevier B.V. All rights reserved.

### 1. Introduction

The 0–1 principle introduced by Knuth [4] states that a comparator network is a sorting network if and only if it sorts all binary inputs. Sorting is not the only functionality that comparator networks are useful for. Additional functionalities include merging two sorted vectors, halving (i.e., separating  $2j$  keys into the  $j$  lowest keys and the  $j$  highest keys), and sorting restricted sets of vectors. For each of these functionalities, some variant or another of the 0–1 Principle [4] was used for proving the correctness of the networks in question.

By the 0–1 Principle and its many variants, comparator networks “work properly” for all valid vectors if and only if they “work properly” for all binary valid vectors. For example, to verify the functionality of a sorting network with  $n$  inputs and outputs, it is suffice to test all  $2^n$  binary vectors. Clearly, there is no need to test constant vectors (vectors, all of whose keys are equal); hence, sorting can be verified by  $2^n - 2$  vectors. Can sorting be verified by fewer binary vectors? A second question lifts the binary restriction and asks: Can sorting be verified by even fewer vectors? Similar questions can be asked for other functionalities.

This paper discusses four functionalities: Sorting, merging, bitonic<sup>1</sup> sorting and halving. The first three functionalities are similar — they all require the output to be sorted. The last functionality, halving, is significantly different from the former ones and this reflects in our proofs and in our results.

We refer to a set of vectors that verifies a specific functionality as a *conclusive set*. So far, only binary vectors were considered for conclusive sets. We introduce the usage of unrestricted vectors (e.g., vectors of natural numbers) for conclusive sets. Interestingly, our main result is that smaller conclusive sets are possible if unrestricted vectors are allowed. In addition, we prove lower bounds on the size of conclusive sets that imply the optimality of our constructions.

Table 1 summarizes our results. The first column in the table lists the four functionalities in question. (See Section 2.1 for formal definitions.) The second column lists the minimal sizes of binary conclusive sets. The third column lists the minimal sizes of unrestricted conclusive sets and the fourth column lists the type of vectors used in our unrestricted conclusive sets

\* Corresponding author. Tel.: +972 4 8293854.

E-mail addresses: [guy@eng.tau.ac.il](mailto:guy@eng.tau.ac.il) (G. Even), [levyt@cs.technion.ac.il](mailto:levyt@cs.technion.ac.il) (T. Levi), [litman@cs.technion.ac.il](mailto:litman@cs.technion.ac.il) (A. Litman).

<sup>1</sup> ‘Bitonic sorting’ means sorting all the bitonic vectors of a certain width. The term ‘bitonic’ is defined in Section 2.1.

**Table 1**  
Summary of results: sizes of conclusive sets for various functionalities.

	Minimal size of binary conclusive set	Minimal Size of unrestricted conclusive set	Type of vectors in our minimal unrestricted conclusive set
Sorting	$2^n - 2[4]$	$\binom{n}{\lceil n/2 \rceil}$	covering vectors
Merging	$\left(\frac{n}{2} + 1\right)^2 - 2[12]$	$\frac{n}{2} + 1$	sandwiches
Bitonic Sorting	$(n - 1) \cdot n[13]$	$n$	unitonic permutations
Halving	$\binom{n}{n/2}$	$\binom{n}{n/2}$	balanced vectors

of minimal size. In this table, as well as in the rest of this paper,  $n$  denotes the *width* of the network in question – the number of keys processed by the network. In the cases of merging and halving,  $n$  is required to be even.

Consider the second column of Table 1. For the first three rows, the corresponding conclusive sets are simply all the valid binary vectors, except the constant ones. The fact that these sets are conclusive is already known and references are given in the table. However, the fact that these sets are of minimal size is a contribution of this paper. Note that the last entry in this column, concerning halving, is significantly different from the first three entries. It is much smaller than the set of all binary vectors which are valid and non-constant. All the above binary conclusive sets share the following property. They are minimal in a very strong sense – each of them is a subset of any binary conclusive set for the same functionality.

Note that, unlike the binary case, the unrestricted conclusive sets of minimal cardinality are not unique; it will be apparent that the same functionality may have several conclusive sets of minimal cardinality that are substantially different.

Consider the ratio between the minimal size of binary conclusive set and the minimal size of unrestricted conclusive set. As discussed in Section 5, this ratio cannot exceed  $n - 1$ , for all functionalities addressed in this paper. We point to the two extreme cases of bitonic sorting and halving. In the case of bitonic sorting, this ratio is  $n - 1$ . On the other hand, for the functionality of halving, this ratio is 1; that is, no improvement is achieved by using unrestricted conclusive sets.

The main motivation for compact and elegant conclusive sets is simplification of the design and analysis of comparator networks. We know of two examples in which conclusive sets were used to construct new networks with useful properties.

The first example concerns fast bitonic sorters of arbitrary width (not a power of two) whose depth is at most  $\lceil \log(n) \rceil + 3$ . Such networks are constructed in [8] using the conclusive set of unitonic permutations introduced in Section 4.

The second example we know of concerns merging networks of minimal depth in which several of the outputs are accelerated. That is, they are generated much faster than the other outputs. Such networks are constructed in [7] using the elegant and compact conclusive set of sandwiches, presented in Section 4.1. A key lemma of this construction, which is proved using sandwiches, is the following lemma:

**Lemma 1.** *Let  $a = \langle a_0, a_1, \dots, a_{j-1} \rangle$  and  $b = \langle b_0, b_1, \dots, b_{j-1} \rangle$  be two sorted sequences of length  $j$ ; let  $0 < k < j$  and let  $c = \langle \max(a_0, b_{k-1}), \max(a_1, b_{k-2}), \dots, \max(a_{k-1}, b_0), \min(a_k, b_{j-1}), \min(a_{k+1}, b_{j-2}), \dots, \min(a_{j-1}, b_k) \rangle$ . Then the sequence  $c$  is bitonic.*

This lemma can be proved using (a variant of) the 0–1 Principle but this leads to many special cases which need to be verified. On the other hand, as demonstrated in [7], sandwiches provide a compact proof, having only two symmetric cases.

An additional benefit of small conclusive sets concerns Black Box testing of the functionality of a given network. In such a test, all members of a conclusive set are fed into the network and the resulting vectors are examined. In this context, the conclusive sets should be as small as possible.

*Previous work.* The main application of the 0–1 Principle is to simplify the design and proof of correctness of sorting and merging networks. We review some of the applications of the 0–1 Principle from the literature. Miltarsen et. al. [12] and Liszka and Batcher [10] used some variant of the 0–1 Principle to prove the correctness of a certain merging network. Bender and Williamson [2] used it to prove structure theorems for recursively constructed merging networks. Batcher and Lee [5] used it to prove the correctness of a  $k$ -merger network whose input consists of  $k$  sorted vectors of equal length. Nakatani et. al. [13] used it to prove the correctness of a bitonic sorter.

*Organization.* This paper is organized as follows. In Section 2, comparator networks are formally defined and various functionalities of comparator networks are presented. In Section 3 the well-known 0–1 Principle for sorting networks is presented along with some variants. These variants enable extending the 0–1 Principle to the functionalities presented in Section 2.1. In Section 4 we present smaller conclusive sets for each of these functionalities. In Section 5 we prove lower bounds on the sizes of binary and unrestricted conclusive sets. These lower bounds match the upper bounds presented in Section 4.

## 2. Comparator networks

A *comparator* is a combinational device that sorts two keys. Namely, it has two incoming edges and it receives a key from each one of them. It has two outgoing edges of distinct types; a **min** edge and a **max** edge. It transmits the minimal key on the **min** edge and the maximal key on the **max** edge.

A *comparator network* (a.k.a. a *network*) is an acyclic network of these devices. See Fig. 1. In this figure, comparators are denoted by circles, a **Min** edge is indicated by a hollow arrowhead, a **Max** edge is indicated by a solid arrowhead and an

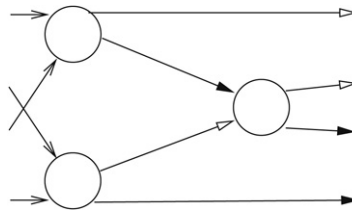


Fig. 1. A merging network of width 4 and depth 2.

input edge is denoted by an open arrowhead. Such a network receives a vector (sequence of keys) via its *input edges* and produces a vector on its *output edges*.

Clearly, a network has the same number of input edges and output edges. To specify how an input sequence should be fed into the network, the input edges are arranged as a sequence. Similarly, the output edges of a network are arranged as a sequence to specify how the network's output is interpreted as a sequence. Sometimes, the order of the input edges and the order of the output edges are implied from the drawing (e.g., from top to bottom).

These networks are useful for performing operations such as merging or sorting on keys – members of a certain ordered set  $\mathcal{K}$ . For most applications, the exact nature of the keys and their number is not important, but this paper is an exception. For most functionalities in question, a minimal conclusive set, for a network processing  $n$  keys, requires  $|\mathcal{K}| \geq n$ . A similar phenomena is shown in [6]. That is, there are some interesting results which are sensitive to the size of  $\mathcal{K}$ . For definiteness, we henceforth assume that  $\mathcal{K} = \mathbb{Q}$  where  $\mathbb{Q}$  is the set of rational numbers.

### 2.1. Functionality

A vector  $v = \langle v_0, v_1, \dots, v_{n-1} \rangle$  is a sequence of keys. The *width* of  $v$  is its length,  $n$ , and is denoted by  $|v|$ . Such a vector can be fed into a network if it is of the appropriate width. As said, the input edges of a network are arranged as a sequence; this specifies how a vector is fed into the network. Similarly, the output edges of a network are also arranged as a sequence; this specifies how the network's output is interpreted as a sequence.

This work discusses four functionalities and each of them is associated with a certain set of *valid* input vectors. To this end, we define the following type of vectors. A vector  $w = \langle w_0, \dots, w_{n-1} \rangle$  is *ascending* (a.k.a. *sorted*) if  $w_i \leq w_j$  whenever  $i \leq j$ . Similarly,  $w$  is *descending* if  $w_i \geq w_j$  whenever  $i \leq j$ . A vector is *ascending–descending* if it is a concatenation of an ascending vector and a descending vector. A vector is *bitonic*<sup>2</sup> if it is a rotation of an ascending–descending vector.

Note that valid inputs to merging networks are different from valid inputs of other types of networks; namely, they naturally consist of two sequences rather than a single one. Hence, we extend the concept of a vector. A vector may be not only a single sequence but also an ordered pair of sequences. A vector  $v = \langle a, b \rangle$  is *bisorted* if the two sequences,  $a$  and  $b$ , are of equal width and are sorted. Note that the width of  $v$  equals  $|a| + |b|$ . Bisorted vectors (of the appropriate width) are the valid input vectors of a merging network. By our definition, the input of a comparator network is a single sequence rather than two sequences. To overcome this, a bisorted vector is combined, in a fixed manner, into a single sequence so it can serve as an input of a comparator network; this can be done, for example, by concatenation.

The four functionalities discussed in this paper are as follows:

sorting: A *sorting network* is a network that sorts all its input vectors.

bitonic sorting: A *bitonic sorter* is a network that sorts all its bitonic input vectors.

merging: A *merging network* is a network that sorts every bisorted vector.

halving: A *halver* receives an even number of keys, and separates them into two sets of equal size. One set contains the lowest keys and the other set contains the highest keys. To this end, the output edges are divided into two (equal size) sets. For example, this division may conform to the order of the output edges as follows. The initial half of the output edges transmit the lowest keys, while the second half of the output edges transmit the highest keys. Following this convention, we say that a vector of even width  $n$  is *halved* if every key in the first  $n/2$  positions is lower or equal to any key in the last  $n/2$  positions.

## 3. The 0–1 Principle

A 0–1 vector (*binary vector*) contains only the keys 0 and 1. The classical 0–1 Principle concerns sorting networks and is as follows:

**Theorem 2** (The 0–1 Principle, [4]). *A comparator network is a sorting network if and only if it sorts every 0–1 vector (of the appropriate width).*

<sup>2</sup> The term 'bitonic' was coined by Batchier [1] and we follow his terminology. We caution the reader that some authors use the same term with other meanings.

The next lemma ([Lemma 4](#)) is a stronger variant of the 0–1 Principle that concerns a single vector rather than all the vectors. Namely, it says that a network sorts a vector  $v$  if and only if it sorts a certain set of 0–1 vectors associated with  $v$ . The lemma uses the following notation.

**Definition 3.**

- A vector  $y$  is an image of a vector  $x$  (or  $x$  covers  $y$ ) if  $|y| = |x|$  and  $x_i \leq x_j$  implies that  $y_i \leq y_j$ , for every two indexes  $i$  and  $j$ .
- A set of vectors  $X$  covers a set of vectors  $Y$  if every member of  $Y$  is covered by some member of  $X$ .

For example, consider the vectors:  $a = \langle 1, 9, 5 \rangle$ ,  $b = \langle 2, 9, 2 \rangle$ ,  $c = \langle 4, 8, 4 \rangle$  and  $d = \langle 7, 7, 7 \rangle$ . Each of these vectors covers all the following vectors. Among these vectors there is exactly one pair ( $b$  and  $c$ ) in which every vector covers the other vector.

The binary relation “ $y$  is an image of  $x$ ” is reflexive and transitive but is neither symmetric nor antisymmetric, as demonstrated in the above examples. A vector  $y$  is a 0–1 image of  $x$  if it is binary and an image of  $x$ . Straightforward 0–1 arguments imply the following two lemmas.

**Lemma 4.** A comparator network sorts a vector if and only if it sorts all its 0–1 images.

**Lemma 5.** A comparator network halves a vector if and only if it halves all its 0–1 images.

We now state 0–1 Principles for merging networks, bitonic sorters, and halvers. These results are known, can be proved by straightforward 0–1 arguments and are used, for example, in [[12,13,4](#)].

**Theorem 6.** 1. A network is a merging network if and only if it sorts every 0–1 bisorted vector.

2. A network is a bitonic sorter if and only if it sorts every 0–1 bitonic vector.

3. A network is a halver if and only if it halves every 0–1 vector.

[Theorem 6](#) is already known; it follows from [Lemmas 4, 5](#) and the following lemma.

**Lemma 7.** A vector is sorted/bitonic/halved if and only if all its 0–1 images are sorted/bitonic/halved.

**Proof.** The hardest case is the bitonic one and we address only this case. The left to right implication is trivial. To prove the converse direction, we show that every non-bitonic vector  $v$  has a non-bitonic 0–1 image. It is not hard to see that  $v$  has a subsequence  $v'$  of length 4 that is not bitonic. (For example, the minimal key and the maximal key of  $v$  are members of  $v'$ ; they divide  $v$  into two parts which are supposed to be ascending and descending. The other two keys of  $v'$  demonstrate that one of these parts is not in the correct order.) Next, pick a rational key  $q$  which is greater than two members of  $v'$  and is smaller than the other two.

Using  $q$  as a threshold, project the rational numbers into the set  $\{0, 1\}$  as follows. Keys lower than  $q$  are mapped to 0 and other keys are mapped to 1. This projection produces a 0–1 image of  $v$  which has a subsequence of length 4 that is not bitonic; therefore, this 0–1 image of  $v$  is not bitonic. ■

#### 4. Smaller conclusive sets

As said, each functionality is associated with a set of valid input vectors. For example, the bitonic vectors are the valid inputs of bitonic sorters.

**Definition 8.** A set of vectors  $C$  is conclusive for sorting/merging/bitonic sorting/halving if every network that “works properly” for every input vector of  $C$ , “works properly” for every valid input vector.

By definition, for every functionality, the set of all valid input vectors is a conclusive set (e.g., the set of all bitonic vectors is conclusive for bitonic sorting). Our goal is to present minimal conclusive sets for these functionalities. We now define several sets of binary vectors, one for each of the first three functionalities.

- Let  $B^{sort}$  be the set of all 0–1 vectors which are non-constant.
- Let  $B^{bitonic} \subset B^{sort}$  be the set of all 0–1 vectors which are non-constant and bitonic.
- Let  $B^{merge}$  be the set of all non-constant 0–1 bisorted vectors.

In other words, each of the sets  $B^{sort}$ ,  $B^{bitonic}$  and  $B^{merge}$  is the set of all binary non-constant vectors that are valid for the corresponding functionality. Each of these sets is conclusive for the appropriate functionality. [Section 4.3](#) presents a binary conclusive set for halving whose definition is substantially different from the above conclusive sets. In [Section 5](#), we show that each of these four conclusive sets is minimal in a very strong sense – it is a subset of every binary conclusive set for the corresponding functionality. [Lemma 4](#), and the fact that the above sets are conclusive, imply the following lemma which is our main tool for constructing even smaller conclusive sets.

**Lemma 9.** Any set of vectors that covers  $B^{sort} / B^{merge} / B^{bitonic}$  is conclusive for sorting/merging/bitonic sorting.

##### 4.1. Sandwiches for merging

In order to describe a small conclusive set for merging we use the following notation. A non-repeating vector is a vector in which each key appears at most once. A permutation is a non-repeating vector of length  $n$  containing all the keys in the set  $\{0, 1, \dots, n-1\}$ . Recall that a bisorted vector is an ordered pair of ascending sequences of equal width. We now define bisorted vectors of a special form called sandwiches.

**Definition 10.** A sandwich is a bisorted vector  $\langle x, y \rangle$  which is a permutation and in which the range of the  $x$  sequence is an interval.

For example the vector  $\langle \langle 1, 2, 3, 4 \rangle, \langle 0, 5, 6, 7 \rangle \rangle$  is a sandwich. The term “sandwich” follows from the fact that the vector can be sorted by inserting the first sequence consecutively in a certain place in the second sequence. Clearly, there are exactly  $n/2 + 1$  sandwiches of width  $n$ . The following lemma is a straightforward observation.

**Lemma 11.** The set of sandwiches covers  $B^{\text{merge}}$ .

Lemmas 9 and 11 imply the following result:

**Lemma 12** (The Sandwich Lemma). The set of sandwiches is conclusive for merging.

#### 4.2. Unitonic vectors for bitonic sorting

Recall that a bitonic sequence is a rotation of an ascending–descending sequence. Similarly, we have the following definition:

**Definition 13.** A unitonic vector is a rotation of an ascending sequence.

Clearly, for every  $n$  there are  $n$  unitonic permutations of width  $n$ . The following lemma is a straightforward observation.

**Lemma 14.** The set of unitonic permutations covers  $B^{\text{bitonic}}$ .

Lemmas 9 and 14 imply the following result:

**Lemma 15.** The set of unitonic permutations is conclusive for bitonic sorting.

#### 4.3. Balanced vectors for halving

Recall that the binary sets  $B^{\text{sort}}$ ,  $B^{\text{merge}}$  and  $B^{\text{bitonic}}$ , were shown to be conclusive for sorting, merging and bitonic sorting. In this section we present the appropriate binary conclusive set for halving. To this end, we say that a 0–1 vector is *balanced* if it contains the same number of zeros and ones.

- Let  $B^{\text{half}}$  be the set of balanced 0–1 vectors.

This section proves that  $B^{\text{half}}$  is conclusive for halving. Later on (Section 5),  $B^{\text{half}}$  is shown to be the minimal binary conclusive set for halving.

##### 4.3.1. Agreeing vectors

To address the issue of halving we need an additional tool which was not needed for the other functionalities. Namely, the concept of agreement.

**Definition 16.** Two vectors,  $x$  and  $y$ , agree if  $|x| = |y|$  and there are no indexes  $i$  and  $j$  such that  $x_i < x_j$  and  $y_i > y_j$ .

For example, consider the vectors:  $a = \langle 3, 3, 3 \rangle$ ,  $b = \langle 2, 5, 5 \rangle$ ,  $c = \langle 4, 4, 8 \rangle$  and  $d = \langle 6, 7, 6 \rangle$ . Every two of these vectors agree except  $c$  and  $d$ . Note that the binary relation “ $x$  and  $y$  agree” is symmetric and reflexive. By the above examples, this relation is not transitive. Clearly, if one vector is an image of another vector then the two vectors agree. The inverse implication is not necessarily true. (E.g., the vectors  $b$  and  $c$  agree but neither one is an image of the other.) We have the following lemma.

**Lemma 17.** Suppose two vectors  $x$  and  $y$  agree. Let  $N$  be a network of width  $|x|$  and let  $x'$  and  $y'$  be the vectors produced by  $N$  when receiving  $x$  and  $y$ , respectively. Then  $x'$  and  $y'$  agree. ■

**Proof.** It is not hard to see that the lemma holds when  $N$  has a single comparator. By induction on the number of comparators, the lemma holds for any network  $N$ . ■

##### 4.3.2. A conclusive set for halvers

In order to construct a small conclusive set for halving, we use the following lemmas whose proofs are straightforward and, therefore, omitted.

**Lemma 18.** Every vector of even width agrees with some member of  $B^{\text{half}}$ .

For any  $j$ , let  $0^j 1^j$  be the vector composed of  $j$  zeros followed by  $j$  ones.

**Lemma 19.** A vector  $x$ , of length  $n$ , is halved if and only if  $x$  and  $0^{\frac{n}{2}} 1^{\frac{n}{2}}$  agree.

The following lemma is the main result of this section.

**Theorem 20.** *The set  $B^{half}$  is conclusive for halving.*

**Proof.** Let  $N$  be a network that halves all members of  $B^{half}$  of width  $n$ . Let  $x$  be a vector of width  $n$  and let  $x'$  denote the output produced by  $N$ , given input  $x$ . By Lemma 18,  $x$  agrees with some vector  $y \in B^{half}$ . The network  $N$  halves all members of  $B^{half}$ ; in particular, it halves  $y$ , producing the output  $0^{\frac{n}{2}} 1^{\frac{n}{2}}$ . Lemma 17 imply that  $x'$  and  $0^{\frac{n}{2}} 1^{\frac{n}{2}}$  agree. By Lemma 19,  $x'$  is halved, and the lemma follows. ■

Theorem 20 and Lemma 5 imply the following result:

**Lemma 21.** *Any set of vectors that covers  $B^{half}$  is conclusive for halving.*

#### 4.4. Conclusive sets for sorting

This section proves the existence of a conclusive set of size  $\binom{n}{\lceil \frac{n}{2} \rceil}$  for sorting  $n$  keys. However, it does not actually constructs such a set. In fact, we do not know of a canonical and elegant conclusive set, of minimal size, for sorting.

Surprisingly, this section is based on the theory of partially ordered sets and on the seminal theorems of Dilworth [3] and Sperner [11]. We use the following notations. Let  $\mathbb{P} = (F, \preceq)$  denote a partially ordered set (poset). Two elements,  $a$  and  $b$  of  $F$ , are *comparable* if  $a \preceq b$  or  $b \preceq a$ . A *chain* (antichain) of  $\mathbb{P}$  is a subset  $Y \subset F$  such that any two distinct elements of  $Y$  are comparable (not comparable).

**Theorem 22** (Dilworth's Theorem [3]). *Let  $\mathbb{P}$  be a finite partially ordered set. Let  $K$  be the cardinality of the largest antichain of  $\mathbb{P}$  and let  $M$  be the minimal number of chains that cover  $\mathbb{P}$ . Then  $K = M$ .*

Recall that, in the context of merging, a vector is an ordered pair of sequences. However, in this section, a vector is always a single sequence; namely,  $v = \langle v_0, v_1, \dots, v_{n-1} \rangle$ . A 0–1 vector of width  $n$  and a subset of  $\{0, 1, \dots, n-1\}$  are two aspects of essentially the same object. In this section we do not distinguish between these aspects. That is, for a vector  $v$ , the two phrases “ $v_i = 1$ ” and “ $i \in v$ ” are equivalent. Similarly, for two 0–1 vectors of the same width,  $u$  and  $v$ , the phrases “ $u \subset v$ ” and “ $u_i \leq v_i$  for every  $i$ ”, are equivalent.

Let  $n \in \mathbb{N}$  be fixed in the following discussion. We focus on the poset  $\mathbb{P}^n = (\{0, 1\}^n, \subset)$ . Let  $V^n$  be the set of unrestricted vectors of width  $n$ . We next show that every chain is covered by some vector of  $V^n$ . To this end, we use the following notations. Let  $X$  be a subset of  $\{0, 1\}^n$ . For an index  $i$ , let  $X[i]$  be the subset of  $X$  defined by  $X[i] = \{v \mid i \in v \in X\}$ . Let  $\widehat{X} \in V^n$  be defined by  $\widehat{X}_i = |X[i]|$  for every  $i$ . For example, let  $n = 5$  and let  $C = \{\{3\}, \{0, 3, 4\}, \{0, 1, 3, 4\}\}$  be a chain of  $\mathbb{P}^n$ . Then  $\widehat{C} = \langle 2, 1, 0, 3, 2 \rangle$ .

**Lemma 23.** *Every chain  $C$  of  $\mathbb{P}^n$  is covered by  $\widehat{C}$ .*

**Proof.** Let  $i$  and  $j$  be two indexes and consider the two sets  $C[i]$  and  $C[j]$ . Since  $C$  is a chain, one of these sets contains the other. Referring to Definition 3 of cover, assume that  $\widehat{C}_i \leq \widehat{C}_j$ . By the above argument,  $C[i] \subset C[j]$ . In other words, for every  $v \in C$ ,  $v_i = 1$  implies  $v_j = 1$ . That is,  $v_i \leq v_j$ ; hence,  $\widehat{C}$  covers  $v$ . Since this holds for every  $v \in C$ , it follows that  $\widehat{C}$  covers  $C$ . ■

Our construction is based on Sperner's famous theorem.

**Theorem 24** (Sperner's Theorem [11]). *The largest antichain of  $\mathbb{P}^n$  is of size  $\binom{n}{\lceil n/2 \rceil}$ .*

The following lemma is the main result of this section.

**Theorem 25.** *There is a conclusive set of size  $\binom{n}{\lceil n/2 \rceil}$  for sorting vectors of width  $n$ .*

**Proof.** By the theorems of Dilworth and Sperner, there are  $\binom{n}{\lceil n/2 \rceil}$  chains that cover  $\mathbb{P}^n$ . By Lemma 23, each of these chain is covered by a single vector; hence,  $\{0, 1\}^n$  is covered by a set of  $\binom{n}{\lceil n/2 \rceil}$  vectors. By Lemma 9, this set is conclusive for sorting. ■

## 5. Lower bounds for conclusive sets

This section shows that our binary conclusive sets and unrestricted conclusive sets are of minimal sizes. Moreover, it shows that our binary conclusive sets are minimal in a stronger sense – each of these sets is a subset of any binary conclusive set for the same functionality. Our main tool is the next lemma which provides, for any 0–1 vector, a network that identifies this vector in the following sense.

**Lemma 26** (The Identification Lemma). *For every 0–1 vector  $v$  which is not constant, there is a network that sorts all the 0–1 vectors of the appropriate width, except  $v$ .*

**Proof.** The desired network,  $N$ , is depicted in Fig. 2. All squares represent sorting networks of various width. The main idea behind the construction is as follows. The input vector, let us call it  $z$ , is partitioned into two vectors,  $d$  and  $u$ , such that the following holds. For most 0–1 vectors, the pair  $\langle d, u \rangle$  is a *separation* of  $z$  – namely, any key of  $d$  is smaller from or equal to any key of  $u$ . In particular, for a 0–1 vector  $z$ , the pair  $\langle d, u \rangle$  is not a *separation* of  $z$  exactly when  $z = v$ . We refer to this functionality as ‘conditional separation’. A network that performs this ‘conditional separation’ can easily be extended to the desired network by sorting the vector  $d$ , to produce the lowest keys, and sorting  $u$  to produce the highest keys.

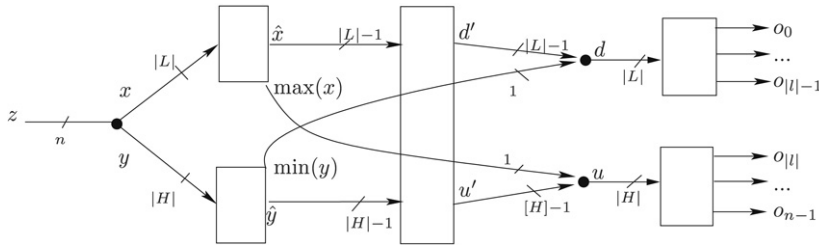


Fig. 2. A network that sorts all 0–1 vectors except  $v$ .

The network  $N$  works as follows. Let  $L$  and  $H$  be the sets of indexes where  $v$  equals zero and one, respectively. Namely,  $L = \{i | v_i = 0\}$  and  $H = \{i | v_i = 1\}$ . Let  $x$  and  $y$  be the subvectors of  $z$  composed of the keys in the positions of  $L$  and  $H$ , respectively. First,  $N$  partitions  $z$  into the vectors  $x$  and  $y$ . (Clearly, this is done without any comparators.) The vector  $x$  is separated (by a sorting network) into its highest key, denoted  $\max(x)$ , and all the other keys, denoted  $\hat{x}$ . Similarly,  $y$  is separated into its lowest key, denoted  $\min(y)$ , and all the other keys, denoted  $\hat{y}$ .

Next, the  $n - 2$  keys of  $\hat{x}$  and  $\hat{y}$  are combined into a single vector; this vector is separated (by a sorting network) into  $\langle d', u' \rangle$  where  $|d'| = |L| - 1$  and  $|u'| = |H| - 1$ . Let  $d$  be the combined vector of  $d'$  and  $\min(y)$ ; similarly, let  $u$  be the combined vector of  $u'$  and  $\max(x)$ . It remains to show that the partition of  $z$  into  $d$  and  $u$  is a 'conditional separation'; that is,  $\langle d, u \rangle$  is a separation of  $z$  if and only if  $z \neq v$ . We consider four cases according to the values of  $\max(x)$  and  $\min(y)$ . It is not hard to see that the case of  $\max(x) = 0$  and  $\min(y) = 1$  holds exactly when  $z = v$ .

- $\max(x) = 0, \min(y) = 1$  : By our construction,  $\max(x)$  is a member of  $u$  and  $\min(y)$  is a member of  $d$ . This implies that  $\langle d, u \rangle$  is not a separation.
- $\max(x) = 1, \min(y) = 0$  : These values, combined with the fact that  $\langle d', u' \rangle$  is a separation, imply that  $\langle d, u \rangle$  is a separation of  $z$ .
- $\max(x) = 1, \min(y) = 1$  : Since  $\min(y) = 1$ , it follows that  $\hat{y}$  is all ones and, therefore,  $u'$  is all ones. Since  $\max(x) = 1$ , it follows that  $u$  is all ones, implying that  $\langle d, u \rangle$  is a separation.
- $\max(x) = 0, \min(y) = 0$  : This case is similar to the previous one. ■

Lemma 26 was sometimes mistaken [14] as proven by Rice [15]. Rice's result does not refer to comparator networks but to a certain set of functions defined by topological means. As shown in [9, Section 3.1], Rice's result is strictly weaker than Lemma 26.

Recall that, by our convention, a halved vector of width  $n$  has the  $\frac{n}{2}$  lowest keys in the first positions and the  $\frac{n}{2}$  highest keys in the last ones. Therefore, every sorted vector is halved. Furthermore, a 0–1 balanced vector is halved if and only if it is sorted. Hence, Lemma 26 has the following corollary.

**Lemma 27.** For every 0–1 balanced vector  $v$ , there is a network that halves every 0–1 vector except  $v$ .

Sections 3 and 4.3.2 show that the sets  $B^{sort} / B^{merge} / B^{bitonic} / B^{half}$  are conclusive for the corresponding functionality. Lemmas 26 and 27 imply that they are minimal in a very strong sense as follows:

**Lemma 28.** A set of 0–1 vectors is conclusive for sorting/merging/bitonic sorting/halving if and only if it contains  $B^{sort} / B^{merge} / B^{bitonic} / B^{half}$ .

We now consider unrestricted conclusive sets. The following lemma states necessary and sufficient conditions for a set to be conclusive for each of the considered functionalities.

**Lemma 29.** A set is conclusive for sorting/merging/bitonic sorting/halving if and only if it covers  $B^{sort} / B^{merge} / B^{bitonic} / B^{half}$ .

**Proof.** Lemmas 9 and 21 provide the left to right implication. Consider the other direction. We focus on the bitonic functionality and the proofs for the other functionalities are similar.

Assume that some 0–1 vector  $z \in B^{bitonic}$  is not covered by a set of vectors  $C$ . By Lemma 26, there exists a network,  $N$ , that sorts all the 0–1 vector except  $z$ . Hence, it sorts all 0–1 vectors covered by  $C$ . By Lemma 4,  $N$  sorts all vectors in  $C$ ; however,  $N$  does not sort  $z$  and therefore,  $N$  is not a bitonic sorter. This implies that  $C$  is not a conclusive set for bitonic sorting.

The proof for halving is similar and is based on Lemma 27 rather than Lemma 26. ■

Clearly, any vector of length  $n$  covers at most  $n - 1$  non-constant 0–1 vectors. This fact and Lemma 29 imply a trivial lower bound on the size of an unrestricted conclusive set. However, these lower bounds are usually not tight, as shown shortly.

We next apply Lemma 29 to show that the unrestricted conclusive sets presented in Section 4 are of minimal size. Clearly, a vector can cover a number of 0–1 vectors; however, there is a class of 0–1 vectors such that every vector covers at most one member of this class. To this end, we extend the term 'balanced' to vectors of odd width as follows. A 0–1 vector of odd width,  $v$ , is *balanced* if it has exactly  $\lceil |v|/2 \rceil$  ones. Clearly, every vector (of odd or even width) covers at most one 0–1

balanced vector. It is not hard to see that:

**Lemma 30.** For every appropriate<sup>3</sup>  $n$ , the sets  $B^{\text{sort}}$ ,  $B^{\text{merge}}$ ,  $B^{\text{bitonic}}$  and  $B^{\text{half}}$  have  $\binom{n}{\lceil n/2 \rceil}$ ,  $n/2 + 1$ ,  $n$  and  $\binom{n}{n/2}$  balanced vectors of width  $n$ , respectively.

Section 4 presents, for every functionality and for every width  $n$ , a conclusive set as follows.

- **Sorting:** a set of  $\binom{n}{\lceil n/2 \rceil}$  vectors.
- **Merging:** the set of  $n/2 + 1$  sandwiches.
- **Bitonic sorting:** the set of  $n$  unitonic permutations.
- **Halving:** the set of  $\binom{n}{n/2}$  balanced 0–1 vectors.

Note that, in the case of sorting, a conclusive set was not constructed; only its existence was proven. In the case of halving the unrestricted conclusive set is, in fact, binary.

Lemmas 29 and 30, and the fact that any vector covers at most one balanced 0–1 vector, imply the following theorem:

**Theorem 31.** Each of the unrestricted conclusive sets in the above list, for the functionalities of sorting/merging/bitonic sorting/halving, is of minimal cardinality.

By Lemma 28, all the functionalities considered in this work have unique 0–1 conclusive sets of minimal cardinality. However, this is not the case for unrestricted conclusive sets. Namely, the same functionality may have several unrestricted conclusive sets of minimal cardinality that are substantially different. For example, consider the functionality of bitonic sorting. As said, the set of unitonic permutations is conclusive for this functionality. Next consider the set of rotations of descending permutations. This set covers  $B^{\text{bitonic}}$  and, by Lemma 9, it is conclusive for bitonic sorting.

## Acknowledgment

We thank Tuvit Etzion for his helpful discussions and suggestions.

## References

- [1] K.E. Batcher, Sorting networks and their applications, in: Proc. AFIPS Spring Joint Computer Conference, vol. 32, 1968, pp. 307–314.
- [2] E.A. Bender, S.G. Williamson, Periodic sorting using minimum delay recursively constructed merging networks, Electron. J. Combin. 5 (1998).
- [3] R.P. Dilworth, A decomposition theorem for partially ordered sets, Ann. Math. 51 (1950) 161–166.
- [4] D.E. Knuth, The Art of Computer Programming vol. 3: Sorting and Searching, second ed., Addison-Wesely, 1998.
- [5] D.I. Lee, K.E. Batcher, A multiway merge sorting network, IEEE Trans. Parallel Distrib. Syst. 6 (1995) 211–215.
- [6] T. Levy, A. Litman, The strongest model of computation obeying 0–1Principles, Technical Report CS-2007-17, Computer Science Department, Technion.
- [7] T. Levy, A. Litman, Accelerating certain outputs of merging and sorting networks, Technical Report CS-2008-10, Computer Science Department, Technion.
- [8] T. Levy, A. Litman, Fast bitonic sorters of arbitrary width (in preparation).
- [9] T. Levy, A. Litman, On merging networks, Technical Report CS-2007-16, Computer Science Department, Technion.
- [10] K.J. Liszka, K.E. Batcher, A Modulo merge sorting network, Symposium on the Frontiers of Massively Parallel Computation, 1992.
- [11] D. Lubell, A short proof of Sperner’s theorem, J. Combin. Theory 1 (1966) 299.
- [12] P.B. Miltersen, M. Paterson, J. Tarui, The asymptotic complexity of merging networks, J. ACM 43 (1) (1996) 147–165.
- [13] T. Nakatani, S.T. Huang, B.W. Arden, S.K. Tripathi, K-way bitonic Sort, IEEE Trans. Comput. 38 (1989) 283–288.
- [14] S. Rajasekaran, S. Sen, A generalization of the 0–1 Principle for sorting, IPL 94 (2005) 43–47.
- [15] W.D. Rice, Continuous algorithms, Topology Appl. 85 (1998) 299–318.

<sup>3</sup> For the functionalities of merging and halving,  $n$  is required to be even.