BOUNDS ON THE SIZE OF TEST SETS FOR SORTING AND RELATED NETWORKS

Moon Jung CHUNG* and B. RAVIKUMAR**

** Department of Computer Science, Michigan State University, East Lansing, MI 48824, USA*
*** Department of Computer Science and Statistics, University of Rhode Island, Kingston, RI 02281, USA*

Received 17 December 1987

We study the size of the smallest test set to decide if a network is a sorting network (sorter), i.e. if it can sort all inputs. Our results include:

(i) The size of the smallest test set to test if a network with n inputs is a sorter is exactly $(2^n - n - 1)$ if 0, 1-inputs are used and exactly $\binom{n}{\lfloor n/2 \rfloor} - 1$ if permutations of $(1\,2\,3 \cdots n)$ are used as inputs.
(ii) The size of the smallest test set to test if a network is a (k, n)-selector is exactly $\sum_{j=0}^{k} \binom{n}{j} - k - 1$ for 0, 1 inputs and $\binom{n}{\min\{\lfloor n/2 \rfloor, k\}} - 1$ for permutation inputs.
(iii) The size of the smallest test set to test if a network is a merging network is exactly $n^2/4$ if 0, 1 inputs are used and exactly $n/2$ for permutation inputs.

1. Introduction

Various models of computation have been proposed to study 'comparison-based problems' such as sorting. A comparator network (network, for short) is one of the models of great significance, studied extensively in the context of parallel sorting (see e.g. [1, 2, 6]). Most of the studies on networks have been of synthetic type: Design a network with some desired properties. In contrast, our work addresses analytic questions of the type: Given an arbitrary network, does it have a given property? Although our primary interest is in solving some combinatorial and complexity theoretic aspects of the problem, we believe that our study will also be useful in testing VLSI circuits for possible hardware failures.

A network has n lines through which n inputs of the network pass and a collection of comparators connecting some pairs of lines such as shown in Fig. 1. (Comparators are shown by vertical lines.) A comparator compares a pair of numbers entering as inputs, compares them and places the smaller number at the top and the larger at the bottom.

The figure also shows the way the network processes the input $(4\,1\,3\,2)$. Suppose we wish to know, based on the input–output behavior of the network, if it is capable of sorting all possible inputs. An exhaustive approach is to test all the $n!$ permutations of $(1\,2 \cdots n)$. However, one can do better by using only

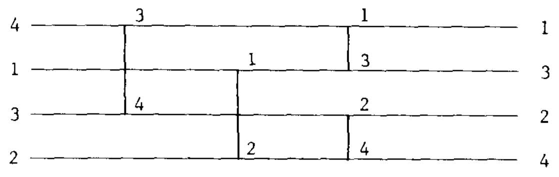

Fig. 1. A compare-interchange network.

0, 1-inputs, thus cutting the number of tests down to 2^n (more precisely, $2^n - n - 1$). This is because of the following result.

Zero-one principle (Knuth). If a network with n input lines sorts all 2^n sequences of 0's and 1's into nondecreasing order, it will sort any arbitrary sequence of n numbers into nondecreasing order.

Our study was originally motivated by the following question: Can the number of tests be reduced further? We show that the answer is 'no' by showing that, for any non-sorted sequence σ of 0's and 1's, there is a network which sorts every sequence except σ. Using this result, we obtain an exact bound on the size of the minimum test set for sorting when the inputs are permutations. We next consider the problem of (k, n)-selection. A network is said to be a (k, n)-selector if it outputs the ith smallest of the inputs at the ith output line for all i, $1 \leqslant i \leqslant k$. We obtain an exact bound on the size of the smallest test set for testing if a network is a (k, n)-selector. We also show that there is a linear size test set to test if a network is a merging network.

It is important to note that, throughout this paper, the comparators used are standard, in the sense of [6]. This means that they cannot be connected or wired upside down, so as to get the larger of the inputs at the top and the smaller at the bottom. Thus, for example, Batcher's bitonic sorter is not a network in our sense. A desirable feature of a network with standard comparators is that, once an input gets sorted, ensuring comparators cannot 'unsort' it. Since our results are primarily lower bounds, this restriction is made only to prove tight bounds. Obviously any of our lower bound results can be immediately carried over to nonstandard networks.

The results presented in this paper have applications in the study of the complexity of decision problems of the following kind:

INSTANCE: A network H.
OUTPUT: 'Yes' if and only if H possesses a given property.

An example of such a decision problem is: "Is a given network a sorting network?". Several decision problems of this kind have been studied by the authors in [3]. Using the result Lemma 2.1, they proved that it is *co*NP-complete to test if a given network is a sorting network. (Rabin [5] proved this result independently by reducing the 3-Dimensional Matching problem.) Further, there is a strong motivation for studying the size of the smallest test sets for various properties since the complexity of testing for a property is closely related to the size of the smallest test set. This relationship is revealed in the following theorem proved in [3].

"For any property for which the size of the smallest test set is at least $c \cdot 2^n$ for all $n \geq 1$ and for some c, $0 < c < 1$, the problem of testing if a given network has that property is intractable, in the sense that the problem is not in P unless $\mathrm{NP} = co\mathrm{NP}$".

Thus to show that a property is intractable to test, it is enough to show that it has a very large test set.

2. Results

A network H of size n is a sequence of pairs of the form $[a_1, b_1][a_2, b_2] \cdots [a_m, b_m]$ where $1 \leq a_i < b_i \leq n$. A network operates on an n-vector of inputs, producing an n-vector of outputs. $[a_i, b_i]$ is a comparator that interchanges the a_ith and b_ith input numbers if they are out of order (i.e., if the a_ith number is larger than the b_ith number.) The network in Fig. 1 can be represented as $[1, 3][2, 4][1, 2][3, 4]$. We consider two types of inputs. (i) n-tuples over $\{0, 1\}$, and (ii) permutations of $(1\,2 \cdots n)$. The following definitions are presented for 0, 1-inputs. These can be extended to permutation inputs in an obvious way. An n-tuple over $\{0, 1\}$ is often referred to as a string in $\{0, 1\}^n$. The output of H on input σ will be denoted by $H(\sigma)$. For a string σ, we denote the substring starting at the ith bit (from left) ending at the jth bit by $\sigma_{i:j}$. The single bit $\sigma_{i:i}$ will be denoted by σ_i. The number of zeroes and ones in σ are denoted by $|\sigma|_0$ and $|\sigma|_1$, respectively. Let H be a network (with n inputs). H is said to be a sorter if, for all $\sigma \in \{0, 1\}^n$, $H(\sigma)$ is sorted. H is a (k, n)-selector if for all $\sigma \in \{0, 1\}^n$, $(H(\sigma))_i$ is the ith smallest bit in σ for all i, $1 \leq i \leq k$. For an even n, a network H with n inputs is said to be an $(n/2, n/2)$-merging network if, for any sorted sequences σ_1 and σ_2 where $|\sigma_1| = |\sigma_2| = n/2$, $H(\sigma_1\sigma_2)$ is sorted.

Let $T \subseteq \{0, 1\}^n$. T is said to be a test set for a network property if, given an arbitrary network H, it is possible to decide if H has the property by observing the outputs of H on inputs taken from T. We are interested in obtaining the size of the smallest test sets for the properties stated above.

Theorem 2.2 shows that it is very hard to test if a given network is a sorter since any test set essentially contains all the strings. The following lemma is useful in proving Theorem 2.2.

Lemma 2.1. *Let σ be an non-sorted string in $\{0, 1\}^n$. There exists a network H_σ such that H_σ sorts all strings except σ.*

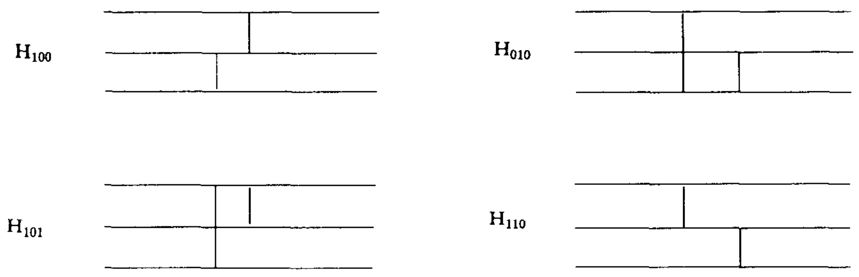

Fig. 2. The case $n = 3$.

Proof. We prove the result by induction on n, the number of input lines. For $n = 2$, the only non-sorted string is 10 and the empty network serves as H_{10}. For $n = 3$, the non-sorted strings are 100, 101, 010 and 110. for each string σ, the corresponding network H_σ is shown in Fig. 2.

In the figures that follow, let $S(i)$ denote an i-input sorting network such as an odd–even merge sorter [2]. We assume that for any non-sorted string σ with $|\sigma| \leqslant n - 1$, there exists a network H_σ such that H_σ sorts all strings of length $n - 1$ over $\{0, 1\}$ except σ and show that the result is true for strings of length n.

Let σ be a string of size n, $n \geqslant 4$. If σ is not sorted, either $\sigma_{1:n-1}$ or $\sigma_{2:n}$ is not sorted. In the remainder of the proof, we consider the former case. Since the latter case is identical, we omit it. Let $\sigma' = \sigma_{1:n-1}$. By induction hypothesis, there exists a network $H_{\sigma'}$ of size $n - 1$ which sorts all inputs except σ'. Since $H_{\sigma'}(\sigma')$ is not sorted, there exist integers k and l $(k < l)$ such that $(H_{\sigma'}(\sigma'))_k = 1$ and $(H_{\sigma'}(\sigma'))_l = 0$. We consider three cases.

Case A. $\sigma_n = 0$ *and* $(H_{\sigma'}(\sigma'))_{n-1} = 0$.

Fig. 3 shows the network H_σ in this case. H_{100} in H_σ has 3 input lines—k, l and n. All other lines bypass H_{100}. (The network H_{100} was presented in Fig. 2.) Let us call the network encased in dotted lines in the figure as H', and the comparator

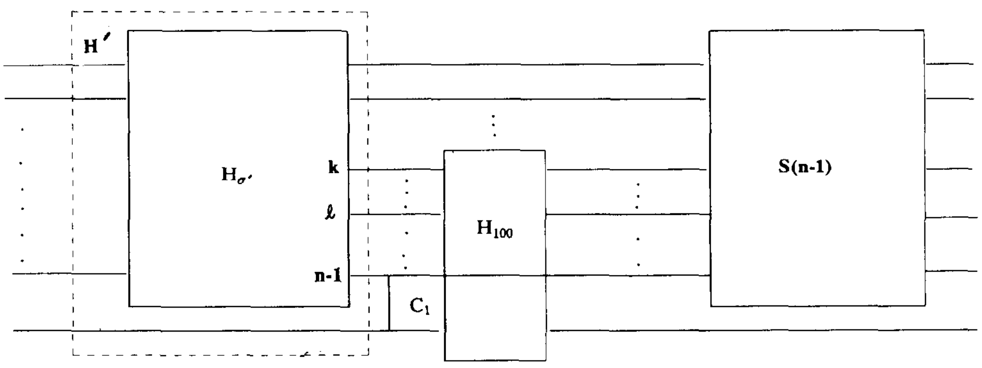

Fig. 3. H_σ for Case A.

immediately following H' as C_1. We first show that H_σ does not sort σ. From the assumptions, we have $(H'(\sigma))_k = 1$ and $(H'(\sigma))_l = 0$. Also, $(H_{\sigma'}(\sigma'))_{n-1} = 0$ implies $(H'(\sigma))_{n-1} = 0$. So, the comparator C_1 does not change the value of line n. This means the network H_{100} receives 100 as input, and thus after passing through H_{100}, the nth line still contains 0. We thus have $(H_\sigma(\sigma))_n = 0$. But $(H_\sigma(\sigma))_{n-1} = 1$. This completes the proof that H_σ does not sort σ.

To show H_σ sorts every string $\tau \neq \sigma$, we consider two cases. In case 1, let $\tau_{1:n-1} = \sigma_{1:n-1}$. Since $\tau \neq \sigma$, we have $\tau_n = 1$. The nth line remains a 1 (since a 1 can never bubble up), while the other $n-1$ bits of input are correctly sorted by $S(n-1)$. In case 2, we have $\tau_{1:n-1} \neq \sigma_{1:n-1}$. In this case $H_{\sigma'}$ will sort $\tau_{1:n-1}$. If $(H'(\tau))_{n-1} = 0$ then $(H'(\tau))_i = 0$ for all $1 \leqslant i \leqslant n-1$, proving the claim. If $(H'(\tau))_{n-1} = 1$, then C_1 and $S(n-1)$ together sort τ.

Case B. $\sigma_n = 0$ *and* $(H_{\sigma'}(\sigma'))_{n-1} = 1$.

Fig. 4 shows the network H_σ in this case. By an argument similar to case A, we can show the result.

Case C. $\sigma_n = 1$.

In this case, the network in Fig. 5 serves as H_σ. Let k be the smallest index such that $((H'(\sigma'))_k = 1$, where H' is as in Fig. 5. In this case, $|\sigma'|_0 \geqslant k$. $S(n-k)$ in the figure is an $(n-k)$ sorter. As shown in the figure, let the comparators between $H_{\sigma'}$ and $S(n-k)$ be named $C_1, C_2, \ldots, C_k$. $(H_\sigma(\sigma))_k = 1$, but $|\sigma|_0 \geqslant k$ thus H_σ does not sort σ. For any $\tau \neq \sigma$, let us show that $H_\sigma(\tau)$ is sorted. Suppose $\sigma_{1:n-1} = \tau_{1:n-1}$. Then $\tau_n = 0$. In this case, the comparator C_k switches the kth and the nth inputs, so that $(H_\sigma(\tau))_k = 0$ for all $j \leqslant k$. The remaining lines $k+1$ through n are sorted by $S(n-k)$. If $\sigma_{1:n-1} \neq \tau_{1:n-1}$, $H_{\sigma'}(\tau_{1:n-1})$ is a sorted string. If $\tau_n = 0$ (otherwise we are done), the k comparators together with the $(n-k)$ sorter will sort τ correctly. □

It can be observed that $H_\sigma(\sigma)$ in each case of the above lemma requires only one more interchange to get sorted.

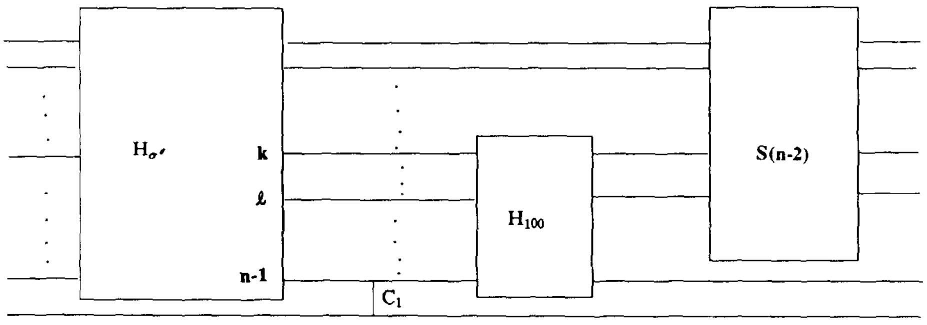

Fig. 4. H_σ for Case B.

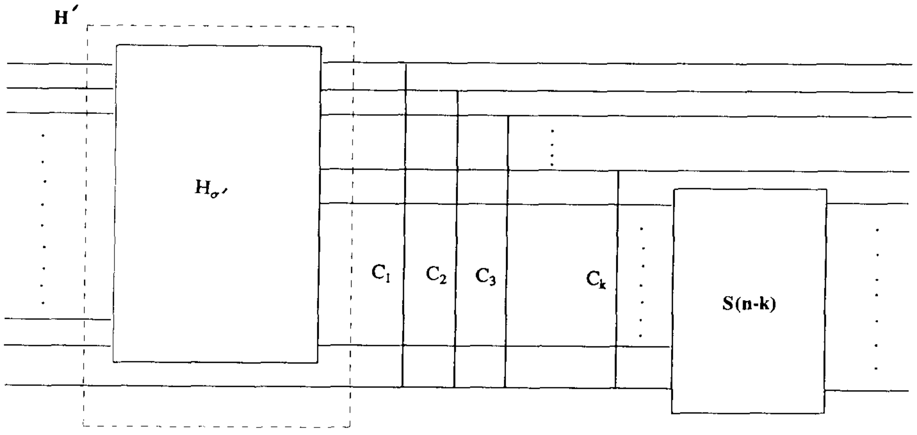

Fig. 5. H_σ for Case C.

It is interesting to note an observation made by Andrew Yao [6] that it is enough to test only $\binom{n}{\lfloor n/2 \rfloor} - 1$ permutations of $(1\,2 \cdots n)$ to decide if a network sorts correctly. The bound $\binom{n}{\lfloor n/2 \rfloor} \sim 2^{n+1}/\sqrt{(2\pi n)}$ (where $\sim$ denotes approximately) is smaller than the size of the test set using only 0's and 1's. The reason for this is that the use of only 0's and 1's makes the behavior of the comparators less sensitive owing to possible duplications in the input, resulting in an overall increase in the size of the test set. We next obtain the size of the test set for sorting when the inputs are permutations. To do this, we need the notion of covering set, defined below, which expresses a close relationship between the two input types used—the set of binary strings and the permutations.

For a permutation $\pi: \{1, 2, \ldots, n\} \rightarrow \{1, 2, \ldots, n\}$, we define a covering set (also called a cover) as the set of binary string obtained by replacing the t largest elements of π by 1, and the others by 0, for all t, $0 \leqslant t \leqslant n$. For example, the cover for $(3\,1\,4\,2)$ is 1111, 1011, 1010, 0010 and 0000. This idea can be extended to the cover of a set of permutations as the union of the covers of the individual permutations in it. Now it is easy to observe that a set of permutations P cannot be a test set for sorting unless its covering set of binary strings is also a test set for sorting. Let D_n denote a set of binary strings of length n, and P_n, a set of permutations of $(1\,2 \cdots n)$. Floyd [6] proved that for any network H, the sets $\{H(x) \mid x \in D_n\}$ and $\{H(x) \mid x \in P_n\}$ can be obtained from the other. (Note that $\{H(x) \mid x \in D_n\}$ is the cover of $\{H(x) \mid x \in P_n\}$.)

Using Lemma 2.1, we show our next result.

Theorem 2.2. *The size of the smallest test set for sorting is* (i) *exactly* $2^n - n - 1$ *if the inputs are string over* $\{0, 1\}$ *and* (ii) *exactly* $\binom{n}{\lfloor n/2 \rfloor} - 1$ *if the inputs are permutations of* $(1\,2 \cdots n)$.

Proof. The claim (i) directly follows from Lemma 2.1 and the zero–one principle. We prove the claimed lower bound when the inputs are permutations. For simplicity we assume that n is even. Consider the set T_1 of binary strings σ of length n with $|\sigma|_0 = n/2$, excluding $0^{n/2}1^{n/2}$. We claim that for any pair of strings σ and σ' in T_1, there is no permutation τ which can cover σ and σ' simultaneously. To show this, let $i_1, i_2, \ldots, i_{n/2}$ be the bits in σ that are 1's. Clearly, not all the bits $\sigma'_{i_1}, \sigma'_{i_2}, \ldots, \sigma'_{i_{n/2}}$ can be 1, lest we should have $\sigma = \sigma'$. Let i_k be such that $\sigma'_{i_k} = 0$. This implies that there is some $j \in \{1, 2, \ldots, n\} - \{i_1, i_2, \ldots, i_{n/2}\}$ such that $\sigma'_j = 1$. Now, if π is a permutation with $\pi(j) > \pi(i_k)$, it cannot cover σ. If $\pi(j) < \pi(i_k)$, it cannot cover σ'. We further know from Lemma 2.1 that every string T_1 is a member of any test set for sorting and thus any test set consisting of permutations for sorting property must be no smaller than $|T_1| = \binom{n}{\lfloor n/2 \rfloor} - 1$, proving the claimed lower bound. The upper bound follows from Andrew Yao's observation. The construction of an optimal test set of permutations is described in [6], Section 6.5.1, Problem 1. □

We next obtain bounds on the test set to test if a network is a (k, n)-selector.

Lemma 2.3. *For $k > 1$, let $T^n_k = \{\sigma \mid$ the length of σ is n, $|\sigma|_0 \leq k$ and σ is not sorted$\}$. Let H_σ be the network constructed in Lemma* 2.1. *For every σ in T^n_k, H_σ outputs the ith smallest input bit in the i output line for $1 \leq i \leq k$, for all inputs except σ.*

Proof. Let σ be in T^n_k. By Lemma 2.1, H_σ produces a sorted output for all inputs $\tau \neq \sigma$ and $H_\sigma(\sigma)$ is not sorted. Let i be the smallest number such that $(H_\sigma(\sigma))_i = 1$. Clearly $i \leq k$. Thus σ is not correctly (k, n)-selected by H_σ. □

We next obtain an upperbound for testing the property "(k, n)-selector". It follows from Lemma 2.3 that the size of the test set for (k, n)-selector property is lower-bounded by $|T^n_k|$. We also show that the set T^n_k is a test set for $\{0, 1\}$ inputs to test if a network is a (k, n)-selector. Using the idea of a cover as in Theorem 2.2, we can also obtain an exact bound for testing a (k, n)-selector when the inputs are permutations.

Theorem 2.4. *The size of the minimum test set to test if a given network is a (k, n)-selector is* (i) *exactly $\sum_{j=0}^{k} \binom{n}{j} - k - 1$ for 0, 1 inputs and* (ii) *exactly $\binom{n}{\min\{\lfloor n/2 \rfloor, k\}} - 1$ for permutation inputs.*

Proof. We first consider 0, 1-inputs. We prove below that T^n_k (defined in Lemma 2.3) is indeed a test set, i.e. if H (an arbitrary n-input network) correctly (k,n)-selects all the inputs in T^n_k, then it is a (k, n)-selector. For two string σ, $\tau \in \{0, 1\}^n$, define a relation $\leq$ as follows: $\sigma \leq \tau$ if and only if for all i, $1 \leq i \leq n$, $\sigma_i \leq \tau_i$. We claim that if $\sigma \leq \tau$ then, for any H, $H(\sigma) \leq H(\tau)$. We can prove this claim by induction on the number of comparators in H using the fact that $a \leq x$ and $b \leq y$ implies $\min\{a, b\} \leq \min\{x, y\}$ and $\max\{a, b\} \leq \max\{x, y\}$. Now let H be a network such that H correctly (k, n)-selects all the strings in T_k^n, and let σ be an arbitrary string in $\{0, 1\}^n$. If $|\sigma|_0 \leq k$, there is nothing to prove, so let $|\sigma|_0 > k$. We can find a string σ' such that $|\sigma'|_0 = k$ and $\sigma \leq \sigma'$ by replacing some of the 0's in σ by 1's. It follows that $H(\sigma) \leq H(\sigma')$ and thus $(H(\sigma))_i \leq (H(\sigma'))_i$ for $i = 1, 2, \ldots, k$. Since $\sigma' \in T_k^n$, H correctly (k, n)-selects σ', i.e. $(H(\sigma'))_i = 0$ for all $i = 1, 2, \ldots, k$. Thus $(H(\sigma))_i = 0$ for all $i = 1, 2, \ldots, k$. Therefore H is a (k, n)-selector. Now (i) follows from Lemma 2.3 and the observation that $|T_k^n| = \sum_{j=0}^{k} \binom{n}{j} - k - 1$.

We next consider permutation inputs. We first obtain a lower bound on the size of test set. Consider two cases:

Case (i). $k < \lfloor n/2 \rfloor$.

First let us show that there is a test set P_k^n of $\binom{n}{k} - 1$ permutations such that P_k^n forms a test set for (k, n)-selector property. Knuth [6] points out (Problem 1 of Section 6.5.1) that for any $k < \lfloor n/2 \rfloor$, there exists a set $B(n, k)$ of $\binom{n}{k}$ permutations such that every t-element subset of $\{1, 2, \ldots, n\}$ appears as the first t elements of at least one of the permutations for $t \leq k$. Let P_k^n be the set of inverse of these permutations excluding the identity permutation. We shall show that P_k^n is a test set for (k, n)-selector property. We first observe that every string in T_k^n is covered by some permutation in P_k^n. We can prove this constructively as follows. Let $\sigma \in T_k^n$ such that σ has a 0 in bit-positions $i_1, i_2, \ldots\ i_t$ where $t \leq k$. By Knuth's construction, there is a permutation $\pi \in B(n, k)$ in which $i_1, i_2, \ldots, i_t$ appear as first t members (not necessarily in this order). It can be seen that π^{-1}, which is in P_k^n, covers σ. P_k^n is a test set for (k, n)-selector follows from the observation that a set of permutations is a test set for (k, n) selector if and only if its cover is a test set for $\{0, 1\}$ inputs.

To prove the lower bound, we first note that every string $\sigma \in T_k^n$ belongs to every test set for (k, n) selector property. Now let $U_k^n = \{\sigma \mid |\sigma|_0 = k,\ \sigma \text{ is not sorted}\}$. Let P be a set of permutations that forms a test set for (k, n)-selector property. Clearly, every string in U_k^n must be covered by some permutation in P. Further, no permutation in P can cover two permutations in U_k^n. Thus the mapping $f : P \rightarrow U_k^n$, defined by $f(x) = y$ if x covers y, is an on-to mapping. Thus $|P| \geq |U_k^n| = \binom{n}{k} - 1$.

Case (ii). $k \geq \lfloor n/2 \rfloor$.

We observe that the set $P_{\lfloor n/2 \rfloor}^n$ of permutations constructed in case (i) covers the set of all binary strings. Thus $P_{\lfloor n/2 \rfloor}^n$ is a test set for (k, n)-selector property. This gives an upper bound of $\binom{n}{\lfloor n/2 \rfloor} - 1$. The lower bound is derived from the fact that no permutation can simultaneously cover any pair of strings in $U_{\lfloor n/2 \rfloor}^n$.

Combining the two cases, we obtain that the size of the smallest test is $\binom{n}{\min\{\lfloor n/2 \rfloor, k\}} - 1$ for permutation inputs. □

We next consider an optimum test set for merging.

Theorem 2.5. *The size of the smallest test set for testing if a network is an* $(n/2, n/2)$*-merging network is* (i) *exactly* $n^2/4$ *if* 0, 1*-inputs are used and* (ii) *exactly* $n/2$ *if permutations are used.*

Proof. It is easy to observe that the set $T = \{\sigma_1\sigma_2 \mid$ lengths of σ_1 and σ_2 are $n/2$, both the strings σ_1 and σ_2 are sorted, and $\sigma_1\sigma_2$ is not sorted$\}$ is a test set for merging. Further, from Lemma 2.1, T can be shown to be the smallest test set. Thus, (i) is proved. To show the sufficiency in (ii), we observe that strings of the form $0^i1^{n/2-i}0^k1^{n/2-k}$ can be covered by $\tau_i = (1\,2 \cdots i\; i+1+n/2 \cdots n\; i+1\; i+2 \cdots i+n/2)$, for a fixed i and arbitrary k. Thus, the set of permutations $\bigcup_{i=0}^{n/2-1}\{\tau_i\}$ forms a test set for merging. To show the necessary part, we exhibit, as in Theorem 2.2, a set $T' \subseteq T$ of strings such that no pair of strings in T' can be covered simultaneously by any permutation. Such a set is $T' = \{0^i1^{n/2-i}0^{n/2-i}1^i \mid 0 \leqslant i \leqslant n/2-1\}$. This completes the proof. □

3. Concluding remarks and suggestions for further research

Our study has revealed some combinatorial properties of networks. We have obtained tight bounds on the size of the test sets for some fundamental properties such as sorting and merging. As mentioned in the introduction, the size of the test set for a property has a close relationship to the complexity of testing a network property and hence deserves further study.

One approach to extending our results is to consider special types of networks and obtain exact bounds on the size of test sets for such special networks. One such restriction is the set of height-k networks defined below. A height-k network is a network such that, for any comparator $[i, j]$ in H, $j - i \leqslant k$. Height-1 networks are called primitive networks [6] and have been studied in [4]. An interesting result shown in [4] is that the size of the smallest set if a height-1 network is a sorter is exactly one, because a height-1 network is a sorter if and only if it sorts the reverse permutation $(n\; n-1 \cdots 2\; 1)$. It would be interesting to obtain exact bounds on the number of tests required to test if a height-2 network is a sorter etc. Another restriction on a network could be based on some uniformity of its structure.

References

[1] M. Ajtai, J. Komlos and E. Szemeredi, Sorting in $C \log n$ parallel steps, Combinatorica 3 (1983) 1–19.
[2] K.E. Batcher, Sorting networks and their applications, Proc. AFIPS Spring Joint Conference (AFIPS Press 1968) 307–314.
[3] M.J. Chung and B. Ravikumar, Strong nondeterministic Turing reduction—a technique for proving intractability, Proceedings of the Second Annual Conference on Structure in Complexity Theory (June 1987) 132–137. (Revised version to appear in J. Comp. and Systems Sciences).
[4] N. de Bruijn, Sorting by means of swappings, Discrete Mathemat. 9 (1974) 333–339.
[5] D. Johnson, NP-completeness column, J. Algorithms 3 (1982) 298.
[6] D.E. Knuth, The art of computer programming, Vol. 3, Sorting and Searching (Addison-Wesley Inc., 1973).