Higher Lower Bounds for the Minimal Depth of n-Input Sorting Networks

Martin Marinov[1], David Gregg[2]

Abstract

We present a new mathematical model for representing comparator networks together with a new algorithm for finding sorting networks of minimum depth. Our algorithm significantly reduces the search space in comparison to the previous state of the art approach. We also provide a computer assisted proof that an eleven-input sorting network must consist of at least eight levels, where previously it was hypothesized to be seven. Since eleven and twelve-input sorting networks of depth eight have been exhibited we conclude that their minimal depth is equal to eight.

Keywords: sorting network, minimal depth, computer assisted proof, extremal sets, extremal sets up to permutation

1. Introduction

A sorting network is an abstract mathematical model designed to sort numbers in a predetermined sequence of comparators. A sorting network consists of n wires and comparators between pairs of wires such that any input of n numbers is sorted by the network, where one wire corresponds to one number. The two most common measures of sorting networks are the total number of comparators — *Bose-Nelson's sorting problem [1]* — and the number of network levels, also referred to as depth. In this paper we extend the knowledge on the problem of empirically searching for sorting networks of minimal depth. The most recent work on the problem dates back to 1989, where Parberry [2] presented a computer assisted proof that no nine-input sorting network of depth six exists.

We take a completely novel approach to exhaustively finding minimal depth sorting networks which significantly reduces the search space when compared to Parberry's approach. When experimentally evaluated our algorithm considers more than 74 times fewer eleven-input comparator networks of depth four than Parberry's method. Our program concluded that an 11-input sorting network of depth seven does not exist. Hence, we provide a computer assisted proof that the minimum depth of an eleven-input sorting networks is eight.

Email addresses: marinovm@tcd.ie (Martin Marinov), dgregg@cs.tcd.ie (David Gregg)

[1]Corresponding author. Dept. of Computer Science, Trinity College Dublin, Ireland. Work supported by the Irish Research Council (IRC).

[2]Lero, Trinity College Dublin.

2. Related Work and Contributions

In Figure 2 we present the current upper and lower bounds for the minimal depth of sorting networks of up to twenty inputs. We now provide background on how we populated the values of this table. We also describe common properties of comparator networks that are used throughout this paper.

van Voorhis [3] provided a mathematical formula for the minimal number of comparators $S(n)$ of an n-input sorting network. The number $S(n)$ is used to derive the least possible depth (presented in Figure 2) of an n-input sorting network by dividing it by the maximum number of comparators per network level.

Knuth [4] showed the optimal depth sorting networks for all $n \leq 8$ described in Figure 2. He also presents the zero-one principle of sorting networks which states that if a comparator network sorts all 2^n binary strings of length n then it is a sorting network.

Baddar [5] worked on heuristic algorithms that are aimed to lowering the upper bound, as per Figure 2, whereas our work is towards increasing the lower bound. Using his technique he found an eighteen-input sorting network of depth eleven which is currently the fastest known one for eighteen inputs.

The most recent relevant work on the problem of finding minimal depth sorting networks dates to more than twenty years ago, where Parberry [2] presented a computer assisted proof for the minimal depth of a nine-input sorting network. He significantly reduced network level candidates for the first two levels, in comparison to the naive approach, by exploiting symmetries of the networks (referred to as first and second normal form [2]). For the remaining network levels he proves that we need to only consider ones with maximal number of comparators. Parberry also found a method, referred to as "The Heuristic", to significantly reduce the search space for the second last level of the network. He used a CRAY super computer to test all nine-input comparator networks of depth six that are in second normal form and pass "The Heuristic" check. He verified experimentally that none of them are sorting networks. It is an immediate consequence of his result that there does not exist a te$n$-input sorting network of depth six.

2.1. Contributions

Our work relates to that of Parberry's [2], in the sense that, our proof for the minimality of the depth of an eleven-input sorting network is a computer assisted one. Nonetheless, we approach the problem from a completely different perspective, namely:

- *Novel mathematical representation of comparator networks* — by their output sets.
- *New algorithm for finding minimal depth n-input sorting networks* — our new algorithm presented in this paper significantly reduces the search space in comparison to Parberry's one [2].
- *Computer assisted proof that the minimal depth of an eleven-input sorting network is eight* — it was previously thought that an eleven-input sorting network of depth seven could exist but had not yet been found. The experimental evaluation of an implementation of our new algorithm proves that no eleven-input sorting network of depth seven exists.

3. Formal Definitions and Known Properties

Definition 3.1. *A* generalized comparator *is an ordered pair* $\langle i, j\rangle$ *such that* $1 \leq i \neq j \leq n$. *A generalized comparator is a* comparator *or* min-max comparator *if* $i < j$. *The values* i *and* j *are referred to as* channels. *A* generalized level L *is a set of generalized comparators such that each channel is involved in at most one generalized comparator, formally if* $\langle a, b\rangle, \langle c, d\rangle \in L$ *then* $|\{a, b, c, d\}| = 4$. *A generalized level is a* level *or* min-max level *if it consists only of (min-max) comparators. The set of all* maximal *(min-max) levels is denoted as* $M(n)$, *as described by Parberry [2]. A* generalized n-input comparator network *is a vector* $\langle L_1, L_2, \ldots, L_d, n\rangle$, *where* $L_1, L_2, \ldots, L_d$ *are levels, and* n *is an positive integer. A generalized* n*-input comparator network is called an* n-input comparator network *if it consists only of (min-max) levels.*

So far we have formally defined the structure of a (generalized) comparator network. We need to define the *output* of applying a comparator network to an *input*, where an input is an n-bit binary string [4]. Applying a network to an input permutes the input vector. Hence, for any fixed input we can define a permutation that models the network behaviour when applied to that particular input.

Notation 3.2. *Denote the set of all permutations of* n *elements as* $\Pi_n = \{\pi : \{1, 2, \ldots, n\} \longmapsto \{1, 2, \ldots, n\} \mid \pi \text{ is bijective }\}$. *Let* $v = \langle a_1, a_2, \ldots\rangle$ *be a vector. Denote by* v_i *the* i*-th coordinate of* v, *namely* $v_i = a_i$.

Definition 3.3. *An* input *is a vector* $\mathbf{x} \in \{0, 1\}^n$ *as per Knuth's [4] zero-one principle. Denote by* I_n *the set of all inputs. The evaluation of an* n*-input comparator network* $C = \langle L_1, \ldots, L_d, n\rangle$ *in channel* i *at level* k *on input* x *is the two dimensional vector* $e_x(i, k)$ *where:*

$$e_x(i,k) = \begin{cases} \langle x_i, i\rangle & k = 0 \\ e_x(i, k-1) & \langle i, j\rangle \in L_k \text{ and } e_x(i,k-1)_1 <= e_x(j,k-1)_1 \\ e_x(j, k-1) & \langle i, j\rangle \in L_k \text{ and } e_x(i,k-1)_1 > e_x(j,k-1)_1 \\ e_x(i, k-1) & \langle j, i\rangle \in L_k \text{ and } e_x(i,k-1)_1 >= e_x(j,k-1)_1 \\ e_x(j, k-1) & \langle j, i\rangle \in L_k \text{ and } e_x(i,k-1)_1 < e_x(j,k-1)_1 \\ e_x(i, k-1) & \text{otherwise} \end{cases}$$

The output of applying C *to* x *is* $V_C(x) = \langle e_x(1, d)_1, \ldots, e_x(n, d)_1\rangle \in I_n$. *The permutation of the coordinates when applying* C *to* x *is* $P_C(x) = \langle e_x(1, d)_2, \ldots, e_x(n, d)_2\rangle \in \Pi_n$.

Intuitively, we say that a vector in I_n is sorted if its values are non-decreasing left-to-right, and a sorting network is one which sorts all possible 2^n input vectors. More formally:

Definition 3.4. *The vector* $\langle x_1, x_2, \ldots, x_n\rangle \in \{0, 1\}^n$ *is* sorted *iff* $x_i <= x_{i+1}$ *for all* $1 \leq i < n$. *A* generalized sorting network *is a generalized* n*-input comparator network for which there exists a permutation* $\pi \in \Pi_n$ *such that* $\pi(V_C(x))$ *is sorted for all inputs* $x \in I_n$. *A* sorting network *is an* n*-input comparator network such that* $V_C(x)$ *is sorted for all inputs* $x \in I_n$.

Theorem 3.5. *For every generalized sorting network there is a sorting network with the same size and depth. If the former has only min-max comparators in the first* k *levels, then the latter is identical in the first* k *levels.*

Proof. See Knuth [4]. □

4. New Theory

We aim to improve the current best known algorithm for finding minimal depth sorting networks. That is for any depth d we need to be able to answer the question of whether there exists a comparator network of depth d that sorts all inputs. Our solution is to perform an exhaustive search of all comparator networks of depth d with pruning. The only way we can reduce the number of candidate networks of depth d is to gain deeper understanding of the structure of comparator networks. All of our newly discovered insights of the structure of comparator networks are based on relations between the *output sets* of comparator networks.

Definition 4.1. *Let the* output set *of a comparator network C be $S_C = \{V_C(x) | x \in I_n\}$. Let the set of all already sorted inputs $T_n = \{(x_1, x_2, \ldots, x_n) \mid x_{i<j} = 0, x_{i>=j} = 1$ for $1 \leq j \leq n+1\}$.*

Definition 4.1 gives us a map $M : C \mapsto S_C$ from a comparator network, defined as levels of comparators, as per Definition 3.1, to an output set. M is not injective, which implies that the number of output sets of comparator networks of depth d is bounded above by the number of comparator networks of depth d. As our goal is to find minimal depth n-input sorting networks it is important to present a method for checking if a comparator network C is a sorting network by only considering its output set S_C.

Theorem 4.2. *An n-input comparator network C is a sorting network iff $|S_C| = n + 1$.*

Proof. If C is a sorting network then S_C must be equal to T_n because every input is sorted by C, hence $|S_C| = n + 1$. It is obvious that $T_n \subseteq S_C$ because $V_C(t) = t$ for any $t \in T_n$. If $|S_C| = n + 1$ then we know that $S_C = T_n$ because $|T_n| = n + 1$ and $T_n \subseteq S_C$, hence C is a sorting network. □

Remark 4.3. *An n-input comparator network C is a sorting network iff $S_C = T_n$.*

In the next section we exploit properties of the output sets of comparator networks which help us to reduce the number of candidates in the search for the minimal depth of an n-input sorting network. Intuitively if two comparator networks have equal output sets then we need to only consider one of them. However there are much stronger properties than equality of output sets.

4.1. Subsets of Output Sets

In this section we present a property of the output sets of comparator networks which links to the problem of finding minimal depth sorting networks. We show that if we can extend the comparator network B to a sorting network by appending l levels to it then we can extend any network A such that $S_A \subseteq S_B$ by appending l levels to it. Hence, if A and B are of the same depth we need to consider only the comparator network A.

Definition 4.4. *Let A and B be n-input comparator networks, where $A = \langle A_1, A_2, \ldots, A_d, n\rangle$, $B = \langle B_1, B_2, \ldots, B_k, n\rangle$, and let L be a level. Define the concatenations $A \oplus L = \langle A_1, \ldots, A_d, L, n\rangle$ and $A \oplus B = A \oplus B_1 \oplus B_2 \oplus \cdots \oplus B_k$. Note that $\oplus$ is associative.*

Theorem 4.5. *Let A, B and C be n-input comparator networks. Suppose that $S_A \subseteq S_B$ and $B \oplus C$ is an n-input sorting network. Then there exists a comparator network C' with the same depth as C such that $A \oplus C'$ is an n-input sorting network.*

Proof. See proof of the more general Theorem 4.8. □

Remark 4.6. *Note that in Theorem 4.5 there is no restriction on the depth of the comparator networks A and B. Moreover, the levels of the comparator networks A and B need not be maximal as per Definition 3.1.*

4.2. Subsets of Output Sets up to Permutation

Knuth [4] has shown that comparator networks are just as powerful as generalized comparator networks. He shows that the group of generalized comparator networks is closed under permutation. Intuitively, we would like to strengthen the result of Theorem 4.5 by considering permutations of output sets. Before we present our new result, we need the following lemma to prove it.

Lemma 4.7. *Let $\pi \in \Pi_n$, $x \in I_n$, and C be a comparator network such that $V_C(\pi(x))$ is sorted. Then $\pi(V_{\pi^{-1}(C)}(x))$ is sorted, where $\pi^{-1}(C)$ is a generalized comparator network.*

Proof. Let $x = \langle x_1, x_2, \ldots, x_n \rangle$ and $P_C(\pi(x)) = \langle p_1, p_2, \ldots, p_n \rangle$. Applying π^{-1} to the equality yields that $P_{\pi^{-1}(C)}(\pi^{-1}(\pi(x))) = P_{\pi^{-1}(C)}(x) = \langle \pi^{-1}(p_{\pi^{-1}(1)}), \ldots, \pi^{-1}(p_{\pi^{-1}(n)}) \rangle$. Applying π to the equality yields $\pi(P_{\pi^{-1}(C)}(x)) = \langle p_1, p_2, \ldots, p_n \rangle = P_C(\pi(x))$. From Definition 3.3 of the functions $V_C(x)$ and $P_C(x)$, we now have that $\pi(V_{\pi^{-1}(C)}(x)) = V_C(\pi(x))$. From the hypothesis we know that $V_C(\pi(x))$ is sorted, hence we conclude that $\pi(V_{\pi^{-1}(C)}(x)) = V_C(\pi(x))$ is sorted. □

Theorem 4.5 tells us that if we can extend the comparator network B to a sorting network by appending l levels to it then we can extend any network A such that $S_A \subseteq S_B$ by appending l levels to it. We now extend this result by weakening the constraint $S_A \subseteq S_B$. We show that it is enough to find one permutation $\pi \in \Pi_n$ such that $\pi(S_A) \subseteq S_B$ to claim that if we can extend the comparator network B to a sorting network by appending l levels to it then we can extend any network A by appending l levels to it.

Theorem 4.8. *Let A, B and C be n-input comparator networks, and $\pi \in \Pi_n$ such that $\pi(S_A) \subseteq S_B$ and $B \oplus C$ is an n-input sorting network. Then there exists a comparator network C' with the same depth as C such that $A \oplus C'$ is an n-input sorting network.*

Proof. From the hypothesis we know that there exists C such that $B \oplus C$ is an n-input sorting network. From $\pi(S_A) \subseteq S_B$ we deduce that $V_C(\pi(x))$ is sorted for all $x \in S_A$ because $\pi(x) \in S_B$. Applying Lemma 4.7 to all $x \in S_A$ and C we deduce that $\pi(V_{\pi^{-1}(C)}(x))$ is sorted. Hence $A \oplus \pi^{-1}(C)$ is a generalized n-input sorting network of depth k. Finally we apply Theorem 3.5 to the generalized sorting network $A \oplus \pi^{-1}(C)$ to show that there exists a comparator network C' with the same depth as C such that $A \oplus C'$ is an n-input sorting network. □

5. New Algorithm for Finding Minimal Depth Sorting Networks

Let us consider all output sets of all n-input comparator networks of depth d and denote them by $E_{n,d}$. Choose only the minimal output sets up to permutation from the family of output sets $E_{n,d}$ and call them $R_{n,d}$. Theorem 4.8 tells us that it is enough to consider only the sets $R_{n,d}$ when searching for sorting networks of minimal depth. For each output set in $R_{n,d}$ we can check if its corresponding comparator network is a sorting network by applying Theorem 4.2. Hence we can answer the question whether there exists an n-input sorting network of depth d. This is clearly enough to find the minimum depth of an n-input sorting network. The pseudo code for answering the question if there exists a n-input sorting network of depth d is presented in Algorithm 1. The algorithm's correctness proof is given by the following lemmas in this section.

Definition 5.1. *Let X be a set of output sets of n-input comparator networks. Define the set of all minimal output sets up to permutation of X as $MinPi(X) = \{S_A \mid S_A \in X : \nexists\, S_B \in X, \pi \in \Pi_n : B < A, \pi(S_B) \subseteq S_A\}$, where by $B < A$ we denote the lexicographic order of networks, as*

described by Parberry [2]. Let the set of all output sets of n-input comparator networks of depth d be defined as $E_{n,d}$. Let the set of all minimal output sets of n-input comparator networks of depth d up to permutation be defined as $R_{n,d} = MinPi(E_{n,d})$.

Remark 5.2. $\forall\ S_B \in E_{n,d}\ \exists\ S_A \in R_{n,d}$ *and* $\pi \in \Pi_n : \pi(S_A) \subseteq S_B$.

From the above remark and Theorem 4.8 we deduce that for depth d it is enough to consider only the minimal output sets up to permutation $R_{n,d}$ within the family of all output sets $E_{n,d}$. Hence, we provide a method to reduce the number of candidate output sets of comparator networks of any fixed depth. In order to construct a practically applicable algorithm for finding minimal depth sorting networks we need to show how to efficiently construct the set $R_{n,d}$, which is not immediately obvious from the definition. The following lemma provides us with a recursive method of constructing $R_{n,d}$ by using $R_{n,d-1}$.

Lemma 5.3. $R_{n,d} = MinPi(X_d)$, *where* $X_d = \{S_{C\oplus L} \mid S_C \in R_{n,d-1}, L \in L_n\}$.

Proof. The set equality of $R_{n,d}$ and $MinPi(X_d)$ follows immediately from the following statement: let S_A and S_B be output sets, L be a level and $\pi \in \Pi_n$ such that $\pi(S_A) \subseteq S_B$ then $\pi(S_{A\oplus\pi^{-1}(L)}) \subseteq S_{B\oplus L}$, recall proof of Lemma 4.7. □

Given the set $R_{n,d}$ we need to be able to answer the question of whether there exists an n-input sorting network of depth d. As discussed, we can apply Theorem 4.2 to every element in the set of output sets $R_{n,d}$ but the following lemma gives us a more practically useful result.

Lemma 5.4. *Suppose that there exists an n-input sorting network of depth d then* $|R_{n,d}| = 1$.

Proof. Let C be an n-input sorting network of depth d and let B be an n-input comparator network. From Remark 4.3 we have $S_C = T_n$. Since $T_n \subseteq S_B$, then $S_C \subseteq S_B$. Therefore $R_{n,d} = \{S_C\} = \{T_n\}$, hence $|R_{n,d}| = 1$. □

5.1. Complexity Analysis

In Figure 1 we present the worst case time and space complexity of every function described by our new Algorithm 1. We now provide details to how we calculate these complexities.

The boolean function *Sortable-In-Two-Levels* is described by Parberry [2] together with its complexity analysis. It is important to note that if the function Sortable-In-Two-Levels(C) returns false then the network C is not sortable by the addition of two levels, and if it returns true then C may or may not be sortable in two levels. The comparator networks we consider are not in first normal form, hence we use the time and space complexity of $\mathrm{O}\big(2^n d\big)$, instead of $\mathrm{O}\big(3^{n/2} d\big)$ [2].

The maximum number of output sets returned by the function *Generate-Next-Depth* is equal to rm, where $r = |R|, m = |M(n)|$. Since the maximum size of an output set is $|I_n| = 2^n$ we can deduce that the space required by the generate function is $\mathrm{O}\big(2^n drm\big)$. For the runtime, the worst case is when $d + 2 = t$. Then for each of the rm output sets we need to test if the corresponding network is sortable in two levels. Hence the worst case runtime is $\mathrm{O}\big(2^n drm\big)$.

The space required by *Min-Sets-Up-To-Perm* is proportional (up to a constant) to the size of the input $\mathrm{O}\big(2^n dr\big)$. As per Definition 5.1 of the function $MinPi(X)$ for every pair of sets $S_A, S_B \in X$ we need to check if there exists $\pi \in \Pi_n$ such that $\pi(S_A) \subseteq S_B$. Hence we need to perform $r^2 n!$ tests of whether S_A is contained in or equal to S_B, which can be done in $\mathrm{O}\big(2^n d\big)$. Since $2^n <= n!$ for all $n \geq 4$ we deduce that the worst case runtime for finding all the minimal up to permutation output sets is $\mathrm{O}\big(n! dr^2\big)$.

For the worst case scenario of the function *Exists-Sorting-Network* we assume that Sortable-In-Two-Levels returns true for every input, and that Min-Sets-Up-To-Perm(X) returns X for every possible set of output sets X. Hence $|R[d]| = m^d$. We obtain the worst case space and runtime bounds by simply substituting r for m^d in the complexities of Generate-Next-Depth and Min-Sets-Up-To-Perm and taking the maximum.

6. Experimental Evaluation

In this section we describe in detail the evaluation of the function Exists-Sorting-Network for $n = 11$ and $d = 7$. Summary of the intermediate steps of the execution of our C++ implementation variations of Algorithm 1 is presented in Figure 3. The program concluded that there does not exist an eleven-input sorting network of depth seven. As an immediate consequence, we deduce that the minimal depth of an eleven-input sorting network is eight. Hence we show that the minimal depth lower bounds of sorting networks with eleven to eighteen inputs must be higher then seven, which is reflected in Figure 2.

Experimental Setup. We performed our experiments on a cluster machine with 128 nodes. Every node of the cluster has 2 sockets of Opteron CPUs clocked at 2.30GHz each and 16GB of main memory. In addition to the cluster we also used a machine with 128GB of main memory for calculations that benefit from large amounts of data fitting into main memory.

Algorithm Variations. We implemented two variations of Algorithm 1. The first one - **Marinov - Subsets Up to Π** - is exactly as described in Section 5 and as presented in Algorithm 1 which reduces the output set candidates at each depth by identifying the minimal subsets up to a permutation within this family. The second variation - **Marinov - Subsets** - identifies only the minimal subsets within the set of candidates, hence eliminating fewer candidates at each depth. The second variation is implemented by changing line 6 in function Exists-Sorting-Network of Algorithm 1 to call the function Min-Sets instead of Min-Sets-Up-To-Perm.

Implementation Verification. For all $n < 11$ we have tested our implementation of Algorithm 1 by comparing the result of function Exists-Sorting-Network correspond with the Figure 2. We also modified our implementation of Exists-Sorting-Network to return a sorting network if one exists (by simple backtracking) and made sure that all the sorting networks produced are valid for all $n < 11$. The function Sortable-In-Two-Levels was verified by making sure that it returns true for all *maximal* output sets that are sortable in two levels, for all $n <= 11$. Note that, Sortable-In-Two-Levels is an implementation of "The Heuristic" [2], as described by Parberry it must return true for all sortable networks and either true or false for any other network. The functions Min-Sets and Min-Sets-Up-To-Perm are unit tested.

Comparison to Parberry's Algorithm. The main advantage of our new approach to finding minimal depth sorting networks in comparison to Parberry's [2] is that we consider fewer candidate networks at every level with the exception of the first level. This is clearly seen in Figure 3a at column **P**, **S** and **M** representing the number of networks grouped by depth for Parberry's and our new approaches. In the same table we can also see the ratio between identifying the minimal subsets and the minimal subsets up to permutation at every depth by looking at column **S / M**. For example, if we look at depth four, we see that Parberry considers 1.4×10^{10} comparator networks and **Marinov Up to Permutation** considers a factor of more than 74 times fewer candidates.

Even at depth two, our algorithm manages to reduce the number of candidate networks of depth two compared to the ones of second normal as described by Parberry [2].

Execution Time. The runtime of our C++ implementation of function Exists-Sorting-Network evaluated at $n = 11$ and $d = 6$ took less than 48 hours of wall clock time which is approximately 10000 CPU hours on a large parallel cluster. Approximately a fifth of the time is spent on identifying the 1.98×10^8 minimal output sets up to permutation at depth four. The rest of the computation time is spent on generating the candidates of depth five for which Sortable-In-Two-Levels returns true. The rest of the execution times are insignificantly small in comparison.

7. Conclusion

This paper has extended the knowledge of the structure of comparator networks when constrained to the problem of finding minimal depth sorting networks. Prior to our work, the last published work on the subject was Parberry [2] over twenty years ago. We have invented a new representation of comparator networks and found new properties of them which led to the development of a new algorithm for finding n-input sorting networks of minimal depth. Using an implementation of our new approach we produced a computer assisted proof that the minimal depth of an eleven-input sorting network is eight.

8. Acknowledgements

Almost all calculations were performed on the Lonsdale cluster maintained by the Trinity Centre for High Performance Computing. This cluster was funded through grants from Science Foundation Ireland.

Work supported by the Irish Research Council (IRC).

We would like to thank Andrew Anderson for his help on improving the quality of this paper.

References

[1] R. C. Bose, R. J. Nelson, A sorting problem, J. ACM 9 (2) (1962) 282–296. doi:10.1145/321119.321126.
URL http://doi.acm.org/10.1145/321119.321126

[2] I. Parberry, A computer assisted optimal depth lower bound for sorting networks with nine inputs, in: F. R. Bailey (Ed.), SC, IEEE Computer Society / ACM, 1989, pp. 152–161.

[3] D. C. van Voorhis, An improved lower bound for sorting networks, in: IEEE Transactions on Computers, Vol. C-21, June 1972, pp. 612–613.

[4] D. E. Knuth, The Art of Computer Programming, Volume III: Sorting and Searching, Addison-Wesley, 1973.

[5] S. W. A.-H. Baddar, K. E. Batcher, An 11-step sorting network for 18 elements, Parallel Processing Letters 19 (1) (2009) 97–103.

[6] I. Parberry, Parallel complexity theory, Research notes in theoretical computer science, Pitman, 1987.

Algorithm 1 Pseudo code for our new algorithm for finding minimal depth n-input sorting networks. The minimal depth of an n-input sorting network is the smallest d for which Exists-Sorting-Network(n, d) returns true. Proof of correctness of the presented functions is given by comments.

```
1:  function EXISTS-SORTING-NETWORK(int n, int d)
2:      C_0 ← n-input comparator network of depth 0
3:      R[0].insert(S_{C_0})                          ▷ R[i] is a set of output sets
4:      for depth = 1, 2, ..., d do
5:          R[depth] ← Generate-Next-Depth(R[depth − 1], depth, d)        ▷ Lemma 5.3
6:          R[depth] ← Min-Sets-Up-To-Perm(R[depth])   ▷ Definition 5.1 of MinPi(X)
7:      end for
8:      if |R[t]|= 1 then                                                 ▷ Lemma 5.4
9:          if |S_{R[t][0]}|= n + 1 then
10:             return true                                               ▷ Theorem 4.2
11:         end if
12:     end if
13:     return false
14: end function
15:
16: function GENERATE-NEXT-DEPTH(set of output sets R, int n, int d, int t)
17:     result ← ∅
18:     for all S_C ∈ R do
19:         for all L ∈ M(n) do                                           ▷ Definition 3.1
20:             if d + 2 = t then
21:                 if Sortable-In-Two-Levels(C ∪ L) then                 ▷ "The Heuristic" [2]
22:                     result.insert(S_{C∪L})                            ▷ Lemma 5.3
23:                 end if
24:             else
25:                 result.insert(S_{C∪L})                                ▷ Lemma 5.3
26:             end if
27:         end for
28:     end for
29:     return result                          ▷ result does not contain duplicates
30: end function
```

Function	Memory Worst Case	Runtime Worst Case
Sortable-In-Two-Levels(C)	$\mathrm{O}(nd)$	$\mathrm{O}(2^n d)$
Generate-Next-Depth(R, d, t)	$\mathrm{O}(2^n drm)$	$\mathrm{O}(2^n drm)$
Min-Sets-Up-To-Perm(R)	$\mathrm{O}(2^n dr)$	$\mathrm{O}(n! dr^2)$
Exists-Sorting-Network(n, d)	$\mathrm{O}(2^n dm^{d+1})$	$\mathrm{O}(n! dm^{2d})$

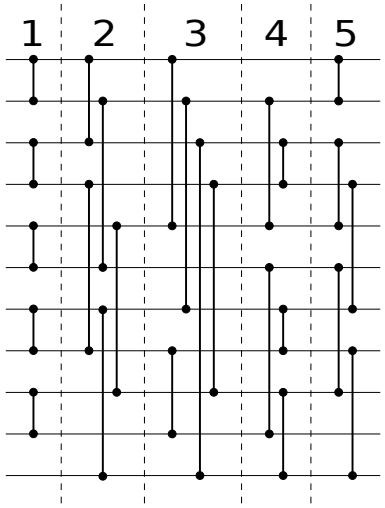
Figure 1: The worst case time and space complexities of all the function used by our new *Algorithm 1* for finding minimal depth n-input sorting networks. We use $r = |R|$ and $m = |M(n)|$ for shorthand writing the complexities. Given that the sorting network verification problem is co-NP complete [6], it is expected that the runtime complexity be exponential.

n	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Upper	1	3	3	5	5	6	6	7	7	8	8	9	9	9	9	11	11	12	12
Lower	1	3	3	5	5	6	6	7	7	**8***	**8***	**8***	**8***	**8***	**8***	**8***	**8***	8	8

Figure 2: Current upper and lower bounds for the depth of an n-input sorting network. Upper represents depth of existing known sorting networks, while lower represents the lower bound on depth of an n-input sorting network. The values marked with star are demonstrated in this paper and were previously equal to seven.

n = 11		**Algorithms**					
		Parberry	**Marinov - Subsets**		**Marinov - Subsets Up to Π**		
		P networks	**S** itemsets	**P / S**	**M** itemsets	**S / M**	**P / M**
Depth	**1**	1	1	1	1	1	1
	2	136	113	1.20	78	1.45	1.74
	3	1,413,720	570,758	2.48	104,667	5.45	13.51
	4	14,695,619,400	737,773,277	19.92	197,175,455	3.74	74.53

(a) Experimental results comparing Parberry's and our new algorithm variations for checking if an eleven-input sorting network of depth seven exists. This table shows for levels one to four the size of the search space considered by every algorithm, for columns **P**, **S** and **M**. We also present the factor of search space size reduction achieved by our new approaches compared to that of Parberry's in columns **P / S** and **P / M**, and comparing our new approaches to each other in column **S / M**.



(b) The only eleven-input comparator network of depth five for which the function Sortable-In-Two-Levels returns true out of all 2.0×10^{12} candidates. Recall that Sortable-In-Two-Levels (our implementation of "The Heuristic" [2]) function can return false positives, but never false negatives. This comparator network turns out to be a false positive, meaning that it can not be extended to a sorting network with the addition of two levels.

Figure 3: Experimental evaluation summary of the function Exists-Sorting-Network-Of-Depth($n = 11$, $d = 7$) which concluded that there does not exist an eleven-input sorting network of depth seven. We present summary of the search space comparison of our approach to Parberry's [2]. Note that we have not implemented Parberry's algorithm since for any depth less than five it is trivial to calculate the number of eleven-input comparator networks that his method would consider [2].