# CORRECTION NETWORKS*

GRZEGORZ STACHOWIAK†

**Abstract.** In this paper we construct sorting comparator networks which correct a fixed number $t$ of faults in a sorted sequence of length $N$. We study two kinds of such networks. One construction yields a fault tolerant unit that attached at the end of any comparator sorting network makes the whole network a sorting one resistant to $t$ passive faults. The second network can be used to 'repair' a sorted sequence in which at most $t$ entries were changed (no fault tolerance is required). The new results of this paper are constructions of comparator networks of depth $1.44 \cdot \log N$ for these problems which is less than the depths of networks described by previous authors [3],[4],[5]. The author believes it is the lower bound for correction networks. The construction of the networks is practical for small $t$. The numbers of comparators used by our networks are shown to be reducible to values optimal up to a constant factor.

**Key words.** sorting network, comparator, fault tolerance, Fibonacci numbers

**AMS subject classifications.** 68Q85, 68W10 (68P10, 11B39)

**1. Introduction.** Sorting is one of the most fundamental problems of computer science. A classical approach to sort a sequence of keys is to apply a comparator network. Apart from a long tradition, comparator networks are particularly interesting due to potential hardware implementations. There are also implementations of these networks as sorting algorithms for parallel computers.

In our approach sorted elements are stored in registers $r_1, r_2, \ldots, r_N$. Registers can be indexed with integers or elements of other linearly ordered sets. In this paper a convenient convention is indexing registers with sequences of integers $\vec{x} = (x_1, x_2, \ldots, x_k)$ ordered lexicographically. By $|\vec{x}|$ we denote the length of $\vec{x}$. A set of all registers having the same first coordinate $x_1$ is called *row* labeled with $x_1$. A set of all registers having the same all but first coordinates we call *column* labeled with the sequence of fixed coordinates. We define operation $\circ$ on sequences of integers as follows:

$$(x_1, \ldots, x_k) \circ (y_1, \ldots, y_l) = (x_1, \ldots, x_k, y_1, \ldots, y_l).$$

A *comparator* $[i : j]$ is a simple device connecting registers $r_i$ and $r_j (i < j)$. It compares the numbers they contain and if the number in $r_i$ is bigger, it swaps them. The general problem is the following. At the beginning of the computations the input sequence of keys is placed in the registers. Our task is to sort the sequence of keys according to the linear order of register indexes applying a sequence of comparators. The sequence of comparators is the same for all possible inputs. We assume that comparators connecting disjoint pairs of registers can work in parallel. Thus we arrange the sequence of comparators into a series of comparator *layers* which are sets of comparators connecting disjoint pairs of registers. The total time needed by a comparator network to perform its computations is proportional to the number of layers of the network called its *depth*.

Much research concerning sorting networks have been done in the past. Their main goals were to minimize the depth and the total number of comparators. The most

*Research supported by University of Wrocław grant 2320/W/IIn/99. This is revised version of the paper: Fibonacci Correction Netoworks [6].

†Institute of Computer Science, University of Wrocław, Przesmyckiego 20, 51-151 Wrocław, Poland (gst@ii.uni.wroc.pl)

famous results are asymptotically optimal AKS [1] sorting network of depth $O(\log N)$ and more 'practical' Batcher [2] network of depth $\sim \frac{1}{2}\log^2 N$ (logarithms in this paper are of base 2). Another well known result is Yao's [7] construction of an almost optimal network to select $t$ smallest (or largest) entries of a given input of size $N$ (*t-selection* problem). His network has depth $\log N + (1 + o(1)) \log t \log\log N$ and $\sim N \log t$ comparators which matches lower bounds for that problem ($t \ll N$).

In this paper we deal with two problems concerning comparator networks. One of them is to construct a comparator network which is a unit correcting $t$ passive faults (see [8]) in any sorting network (some comparators are faulty and do nothing). Such a unit can be attached to any sorting network e.g. AKS (as a number if its last layers) so that the whole network is a sorting one resistant to $t$ faults. The unit has to correct all the faults present in the sorting network and be resistant to all errors present in itself. Such a correcting unit we call *t-tolerance* network. The best result concerning such networks is that of Piotrów [4], who constructed asymptotically optimal $t$-tolerance unit of depth $O(\log N + t)$ having $O(Nt)$ comparators. The exact constants hidden behind these big $O$-h's were not determined, but since Piotrów uses network [5] the constant in front of $\log N$ in $O(\log N + t)$ is at least 2.

The other problem is to sort an almost sorted sequence. Assume we have a large sorted database with $N$ entries. In some period of time we make $t$ modifications of the database and want to have it sorted back. These modifications are swaps between pairs of elements and changes of element values. We design a specialized comparator network of a small depth to 'repair' the ordering and avoid using costly general sorting networks. Such a network to sort back a sorted sequence in which at most $t$ changes were made we call *t-correction* network. The best known general result here is network of Kik, Kutyłowski, Piotrów [3] of depth $4 \log N + O(\log^2 t \log\log N)$.

The networks in [3] [4] are based on a nice construction by Schimmler and Starke [5] of a 1-correction network of depth $2 \log N$ and requiring $3.5N$ comparators. Our goal is to reduce the constant in front of $\log N$ in the depth, which is most essential if $t$ is small and $N$ big. We present $t$-tolerance and $t$-correction networks that for fixed $t$ have depths $\sim \log_{(1+\sqrt{5})/2} N = \alpha \log N$, where

$$\alpha = \frac{1}{\log_2 \frac{1+\sqrt{5}}{2}} = 1.44\ldots.$$

This way our networks have smaller depths than any correction networks described by previous authors.

For sorting networks the following useful lemma called zero–one principle holds:

LEMMA 1.1 (zero–one principle). *A comparator network is a sorting network if and only if it can sort any input consisting only of 0's and 1's.*

This lemma is the reason, why from now on we consider inputs consisting only of 0's and 1's. Below we formulate analogous lemmas for tolerance and correction networks.

We say 0 (1) is *disturbed* if it is changed to 1 (0). Resulting 1 (0) we call *displaced.* A sequence of 0's and 1's produced from a sorted sequence by disturbing at most $t$ zeros and at most $t$ ones we call *t-disturbed.*

Let us remind the proof of zero–one principle. We have an input consisting of arbitrary elements and prove it is sorted. To do it we take an arbitrary $a$ from this input and prove it goes to the register corresponding to its rank in the sequence. We replace all elements bigger than $a$ by 1, and smaller by 0. Indeed the only difference between outputs for the sequence where $a$ is replaced by 0 and for one where $a$ is replaced by 1 is the register with the index corresponding to $\operatorname{rank}(a)$.

Now we deal with an input obtained by $t$ modifications in a sorted sequence. We transform it to a 0-1 sequence as in the proof of zero–one principle. Swapping two elements of an input corresponds to disturbing a 0 and a 1 or doing nothing. Changing element value is equivalent to changing at most one position of 0-1 sequence. Thus an arbitrary input made from a sorted one by $t$ modifications corresponds to a $t$-disturbed 0-1 sequence. This gives us a useful lemma being in fact a characterization of $t$-correction networks.

LEMMA 1.2. *A comparator network is a $t$-correction network if it can sort any $t$-disturbed input.*

Now we try to sort a 0-1 sequence using a faulty sorting network. Passive faults in this network cause some comparators to do nothing, so the network behaves like they were removed. One faulty comparator can cause a single pair of 0 and 1 to change their way in the network. Finally it causes at most one swap between 0 and 1 in the output sequence. This gives us an analogous lemma for $t$-tolerance networks.

LEMMA 1.3. *A comparator network is a $t$-tolerance network if for any $x \leq t$ it can sort any $x$-disturbed input if we remove any set of $t - x$ comparators.*

We define *dirty area* for 0-1 sequences contained in the registers during computations of a comparator network. Dirty area is the minimal set of subsequent registers such that below these registers (in registers with lower indexes) there are only 0's and above there are only 1's. A $t$-disturbed input in which only 0's are disturbed we call *$t$-partially-disturbed.* A comparator network that can reduce dirty area size to at most $\Delta$ for any $x$-partially-disturbed input if $t - x$ comparators are faulty we call *$(t, \Delta)$-partial-tolerance.* Comparator network that can reduce dirty area size to at most $\Delta$ for any $t$-partially-disturbed input we call *$(t, \Delta)$-partial-correction.* For both kinds of networks the output is $t$-partially-disturbed, because a 1 can only increase the index of its register during computations. The final size of dirty area $\Delta$ can be some function $\Delta(N, t)$ of $N$ and $t$.

**2. One Disturbed Position.** In this section we consider sorting inputs with one disturbed position. For simplicity we assume that the input has a single disturbed 0 (i.e. is 1-partially-disturbed). So we have a displaced 1 at this position. We describe a comparator network $F_N$ sorting any such an input of size $N$.

First we recall the definition of Fibonacci numbers $f_k$:

$$f_0 = f_1 = 1,$$

$$f_k = f_{k-2} + f_{k-1}.$$

We define the numbers $\varphi_k, \psi_k$ and $\vartheta_k$ behaving similarly to $f_k$:

$$\varphi_0 = \varphi_1 = \psi_0 = \psi_1 = 1,$$

$$\psi_k = \varphi_{k-2} + \psi_{k-1},$$

$$\varphi_k = \text{ the largest odd number smaller or equal } \psi_k.$$

$$\vartheta_k = 2\psi_k - \varphi_k$$

Define $\mathrm{LG}(n)$ to be the smallest $k$ such that $\varphi_k \geq n$. Let $r_1, r_2, \ldots, r_N$ be registers. The network $F_N$ consists of $d = \mathrm{LG}(N)$ subsequent layers $L_1, L_2, \ldots, L_d$ such that:

$$L_p = \{[2i + p : 2i + p + \varphi_{d-p}] | i \in \mathcal{Z}\}$$

The way we define $L_p$ requires a few words of comment. From all comparators in the definition of $L_p$ only those exist whose end registers are well defined and belong to the set of all registers. This convention is maintained for the rest of the paper. Now we prove that the comparator network defined above corrects a single disturbed position indeed and estimate how many layers it has.

FACT 2.1. $d = \mathrm{LG}(N) \sim \log_{(1+\sqrt{5})/2} N = \alpha \log N$

*Proof.* The Fact follows directly from inequalities $f_{k-1} \leq \varphi_k \leq f_k$ which can be easily proven. □

Any input with a single disturbed position has this *border* between 0's and 1's somewhere inside. We can define the border to be the space between registers containing 0 and 1 when we replace the only displaced 1 by 0. If after replacing all the registers contain 0 (or 1) then the border is above (or below) all the registers. We define the *distance* between displaced 1 and the border to be the number of 0's in registers with higher indexes than one containing displaced 1. Our network proves to reduce this distance very efficiently. It ends computations, when the distance is guaranteed to be 0. The fact the network $F_N$ really corrects a single disturbed 1 is implied by the following lemma.

LEMMA 2.2. *After applying the first $p$ layers of $F_N$ the distance between the single displaced 1 and the border is smaller than $\psi_{d-p+1}$. If displaced 1 is in a register $r_i$ for which $i \equiv p \bmod 2$, then this distance is smaller than $\varphi_{d-p}$.*

*Proof.* We proceed by induction on $p$. For $p = 0$ the lemma is obvious, because $\varphi_d \geq N$. Assume that the lemma holds for $p-1$ and prove it for $p$. Let $i$ be the index of the register containing displaced 1 just before we apply layer $p$. If the displaced 1 is not moved by layer $p$, then two cases are possible. The first is $i \not\equiv p \bmod 2$ and from inductive hypothesis for $p$ the distance is smaller than $\varphi_{d-p+1}$ which is not bigger than $\psi_{d-p+1}$. In the second case $i \equiv p \bmod 2$ the fact 1 is not moved by layer $p$ means that the distance has to be smaller than $\varphi_{d-p}$. If this 1 is moved then its distance is reduced by $\varphi_{d-p}$ from some value smaller than $\psi_{d-p+2}$. Thus after applying layer $p$ this distance is smaller than

$$\psi_{d-p+2} - \varphi_{d-p} = \psi_{d-p+1}.$$

□

By the last lemma at any moment of computations there is a set of registers below the border such that a single displaced 1 contained one of these registers is guaranteed to get to the border. After the first $p$ layers this set consists of all registers $r_i$ below and in the distance from the border smaller than

- $\psi_{d-p+1}$ if $i \not\equiv p \bmod 2$
- $\varphi_{d-p}$ if $i \equiv p \bmod 2$

We call this set the *correction area.*

**3. Partial Tolerance Network.** In this section we describe a $(t, t^2+t)$-partial-tolerance network $T(s, N, t)$ of a small depth. The network is constructed for a parameter $s$ being an arbitrary integer constant. Later in this paper we show how having this network we can easily produce a $t$-tolerance network of a similar depth.

The main idea of the construction $T(\cdot)$ is applying a number of subnetworks $F_{N'}$: some of them delayed in comparison to the others. At the beginning of computations displaced 1's of an $x$-disturbed input are moved to least delayed subnetworks (borders between 0's and 1's are in the same place in all subnetworks). As we proved $F_{N'}$ guarantees sorting any input with a single displaced 1. If the number of displaced 1's is bigger, they become to disturb each other to get to the border. At least one of them gets to the border but others do not have to. We solve this problem moving from time to time displaced 1's that drop out of correction area to another subnetwork. This new subnetwork is delayed in comparison to the previous one. This way 1's that lost their chance to get to the border in one subnetwork regain it in another.

Now we describe the whole network in a more formal manner. We use the set of ordered pairs $(i,j)$ for $i \in \{1,\ldots,N/t\}, j \in \{1,\ldots,t\}$ as the set of register indexes. Let us have an $x$-partially-disturbed input. If we change all displaced 1's to 0's, then in each column (see the definition of the column) we have the highest 0 almost in the same row in its column (the row indexes can differ by 1 between columns). Network $T(s,N,t)$ consists of two parts: one preceding another. The first part is preprocessing consisting of the sequence of layers:

$$P_1, P_2, \ldots, P_{3t},$$

where

$$P_q = \{[(i, 2j+q) : (i, 2j+q+1)] | i, j \in \mathcal{Z}\}.$$

The following easy fact describes the role of the first part in the network $T(\cdot)$.

FACT 3.1. *After applying the first part of $T(\cdot)$ to $x$-partially-disturbed input all displaced 1's are pushed to registers with biggest possible coordinate $j$ (if the number of faults in is not bigger than $t-x$). It means that: $r(i,j)$ contains displaced 1 implies $r(i,j+1)$ also contains 1.* □

Let $d = \mathrm{LG}(N/t)$. The second part of the network which does the main work is the sequence of layers:

$$L_1, L_2, \ldots, L_s, C_s, L_{s+1}, \ldots, L_{2s}, C_{2s}, L_{2s+1}, \ldots, L_{3s}, C_{3s}, L_{3s+1}, \ldots$$

where

$$L_p = \{[(2i+p, j) : (2i+p+\varphi_{d-p+2s(t-j)}, j)] | i, j \in \mathcal{Z}\}$$

and

$$C_q = \{[(2i+q+1, j) : (2i+q+1+\varphi_{d-q+2s(t-j)+1}, j-1)] | i, j \in \mathcal{Z}\}$$

For this section we take $\varphi_k = 1$ for $k < 0$. The second part of the network $T(s,N,t)$ has altogether $(1+1/s)\mathrm{LG}(N) + 2(s+1)(t-1)$ layers. As we see the layers $L_p$ are layers of $F_{N/t}$ inside columns. Layers $C_q$ roughly speaking move displaced 1's to the next delayed columns when they are beyond correction areas in their columns.

Now we analyze what happens to an $x$-partially-disturbed input in the network $T(\cdot)$. We assign to each 1 during computations the property of being or not being *active*. Just after the first part we switch each displaced 1 to be active and each not displaced 1 not to be active. We define the parameter $b$ being the index of the row which behaves like the border in $F_{N/t}$. At the beginning of the computations $b$ is the index of the highest row containing only 0's and displaced 1's. An active 1 ceases to be active at the moment it starts to be in a row $i : i \geq b$. At the same moment $b$ is decreased by one which assures that all inactive 1's are always in rows above $b$. We should mention, that there are two ways for a displaced 1 to cease to be active. It can either be moved to a higher row or $b$ can be decreased to the index of its row. If a couple of 1's should simultaneously cease to be active, because they all are in rows with indexes not smaller than $b$, then they do it one by one.

We assign an integer value $v$ to each displaced 1. We first define *destination column index* $u$ of an active displaced 1 in a given moment of computations. To do it we change all other active 1's to 0's, repair all faulty comparators in the network and take $b$ as it is at this moment of computations. Then we continue computations of the network. If the 1 gets to a row $i : i \geq b$, then destination column index $u$ is the index of the column from which comparator moves it to this row. If displaced 1 does not get to such a row at all, then the index $u$ is set to be 0. The current value $v$ of an active 1 is equal to the minimum of all values $u$ assigned to this 1 till the considered moment of computations. When the 1 ceases to be active, its value remains unchanged till the end of computations. It is not hard to see from the definition, that at the beginning of the second part each 1 has value equal to the index of its column, because of the layers of the network $F_{N/t}$ in this column.

The following facts describe behavior of values assigned to 1's:

FACT 3.2. *If an active 1 is stopped by a passive fault, then its value decreases by one or does not change.*

*Proof.* We prove, that its destination column index $u$ does not change or decreases by one. We trace the stopped 1 repairing all the faults it can encounter and changing all other active 1's to 0's as in the definition of $u$. If $u$ obtained this way is equal to the index of the column in which 1 was stopped, then it means that $u$ was not for sure decreased by the fault, because such $u$ is maximal for this column. So we consider the case that $u$ is smaller than this index. Let us now trace the stopped 1 only for $2s + 1$ layers after stop occurred. During the time we have a layer $C_q$ preceded by layer $L_q$ so the 1 decreases the index of its column in this period. Now we analyze what the destination column index $u$ is just after the first layer $C_q$ which moves the 1 to the next column (with decreased index) in comparison to $u$ just before it is stopped by a fault. Just after $C_q$ the column index is smaller by one, the row is higher, and the layer is later by at most $2s + 2$. The next column because of its delay is at the same or earlier phase of its computations, as the column in which the stop occurred was at the moment of the stop. It proves that the index $u$ of 1 does not decrease by more than one. □

FACT 3.3. *Assume an active 1 is stopped by another active 1 and its value decreases. In such a case its value becomes to be not smaller than the value of 1 causing the delay decreased by one.*

*Proof.* There is no difference for a displaced 1 between being stopped by a passive fault and another displaced 1. Just before one active 1 stops another they must have the same destination column index. □

THEOREM 3.4. *Network $T(s, N, t)$ is a $(t, t^2 + t)$-parital-tolerance network of depth*

$$\alpha\left(1 + \frac{1}{s}\right) + O(st).$$

*Proof.* Putting together the facts one can see that at the end of computations 1's which were displaced at the beginning of the second part have values $v_1, v_2, \ldots, v_x$.

Without loss of generality we can assume that the values form a not increasing sequence. Because of the Facts the difference $v_i - v_{i+1}$ is not bigger than the number of faults 1 with $v_{i+1}$ encountered increased by one. Also $v_1 = t$ is not smaller, than $t$ decreased by the number of faults that stopped 1 corresponding to $v_1$. We have not more than $t$ displaced 1's and faults altogether. Thus $v_x \geq 1$ and all 1's are not active at the end of the second part. It gives dirty area size not bigger than $t^2 + t$, because $b$ is decreased $x$ times during the computations. □

**4. Partial Correction Network.** Now we are going to describe a $(t, ct(\log N)^{c_s \log t})$-partial-correction network $C(s, N, t)$, where $s$ is an integer constant and $c_s$ depends on $s$. This network has depth $\alpha(1 + 1/s)\log N + c_s(1 + o(1))\log t \log\log N$. We show later in this paper how from this network we can obtain a $t$-correction network of almost the same depth. For this section we change denotation $\psi_k$ to $\psi(k)$ (the same for $\varphi$ and $\vartheta$). We also put $\vartheta_k = 1$ for $k < 0$.

Before we begin to construct the network $C(\cdot)$ we prove some lemmas about network $F_N$ on which the construction is based.

LEMMA 4.1. *Assume 1-partially-disturbed input is processed by $F_N$. In a given moment of computations the single displaced 1 is in register $i$. There are at most $f_{s+1}$ registers where it could have been $s$ layers before this moment: at most $f_s$ of the same parity as $i + s$ and at most $f_{s-1}$ of the opposite parity.*

*Proof.* It is enough to prove the lemma for the case, where $i$ is the higher register of a comparator just before the given moment of computations (in the other case the 1 waited in $i$ for the last layer). In such case there are at most $2 = f_2$ registers where displaced 1 could be a layer before this moment: at most $1 = f_0$ of parity of $i$ and at most $1 = f_1$ of parity $i + 1$. Now we proceed by induction on $s$. For $s + 1$ layers we have at most $f_s$ registers of parity $i + s$ because displaced 1 in such register is not moved by additional first layer. We can have at most $f_{s+1}$ registers of the opposite parity because there are at most $f_{s+1}$ registers displaced 1 can be in before $s$ layers. For any such a register there is at most one register of parity $i+s+1$ from which this 1 can come by additional first layer. Thus altogether there are at most $f_{s+2} = f_s + f_{s+1}$ registers from which displaced 1 can come to register $i$ by $s + 1$ layers preceding the given moment. □

The next lemma describes behavior of $F_N$ if the number of displaced 1's is bigger than one

LEMMA 4.2. *Assume that in a given moment of computations not less than $t$ displaced 1's of a partially disturbed input are in the correction area of $F_N$. In such a case after next $s$ layers at least $t/f_{s+1}$ displaced 1's remain in the correction area.*

*Proof.* For each register in correction area after $s$ layers there are at most $f_{s+1}$ registers in a given moment from which by these layers a single displaced 1 comes to this register. If there are more than one displaced 1 in those registers, then still one of them comes to this destination register. Thus at least $t/f_{s+1}$ displaced 1's come to the correction area after $s$ layers. □

The main idea of construction $C(\cdot)$ is similar to that of $T(\cdot)$. We have a number of disjoint subnetworks $F_{N'}$. At the beginning all displaced 1's of a $t$-partially-disturbed input are moved to a few subnetworks (other become free from displaced 1's). In each subnetwork the border between 0's and 1's is in the same place. Each $s$ steps displaced 1's that drop out from correction area in their subnetwork are moved to another subnetwork not containing previously any displaced 1's. These moved 1's are in correction area of their new subnetwork, because the new $F_{N'}$ is delayed by $s + 1$ steps. In the delayed $F_{N'}$ there is at most fraction $1 - 1/f_{s+1}$ of displaced 1's from the previous $F_{N'}$. Thus the total delay cannot grow very much because in the subsequent subnetworks maximal numbers of displaced 1's go down exponentially. In fact this idea is similar to that applied in [3]. The modifications consist in applying $F_N$ network and putting $C_q$ layers not every second step, but less frequently. The following simple combinatorial fact says us, that in our construction the number of subnetworks $F_{N'}$ is not too big.

FACT 4.3. *The number of nondecreasing sequences $j_1, j_2, \ldots, j_k$ for $0 \le k \le K$ and $1 \le j_l \le J$ is equal:*

$$\binom{J+K}{K} = O(J^K)$$

*Proof.* The number is the same as the number of nondecreasing sequences of integers $j_l : 0 \le j_l \le J$ of length $K$ which is the same as the number of increasing sequences of integers $j_l : 1 \le j_l \le J + K$. □

Now we define network $C(\cdot)$ in a more formal way. Let $K = -\log_{1-1/f_{s+1}} t$, $J = \lceil \mathrm{LG}(N) \rceil + K$. In the network $C(s, N, t)$ indexes of registers have the form $(n') \circ \vec{j} \circ (J + \tau)$. In this denotation $\tau \in \{1, \ldots, t\}$, $\vec{j} = (j_1, \ldots, j_k)$ is a nondecreasing sequence of integers $j_l \in \{1, \ldots, J\}$ of length at most $K$ and $n' \in \{1, \ldots, N'\}$, where $N' = \frac{N}{\binom{J+K}{K} t}$.

Now we define the layers of the network that reduce dirty area of any $t$-partially disturbed input. Denote by $b$ the index of the highest row containing only 0's and displaced 1's. We treat the space between rows $b-1$ and $b-2$ as the *border* between 0's and 1's for the needs of our algorithm. The *distance* between a row and the border is the number of rows between them. Displaced 1's below the border we call *active*.

The network $C(s, N, t)$ consists of two parts (one preceding another). The first part consists of selectors [7] for the $t$ largest entries in each row of registers. After the first part all active 1's get to registers $R(n', \tau + J)$. Indexes of these registers are lexicographically biggest in each row. This first part has depth $\sim c_s \log t \log\log N$. The constant $c_s$ depending on $s$ is $O\left(\frac{1}{\ln(1-1/f_{s+1})}\right) = O(f_s)$.

Let $d = \mathrm{LG}(N')$. The second part consists of the sequence of layers

$$L_1, L_2, \ldots, L_s, C_s, L_s, L_{s+1}, \ldots, L_{2s}, C_{2s}, L_{2s+1}, \ldots, L_{3s}, C_{3s}, L_{3s+1}, \ldots$$

where

$$\begin{aligned}
L_p &= \{[(2i+p) \circ \vec{j} \circ (\tau + J) : (2i + p + \varphi(d + (s+1)|\vec{j}| - p)) \circ \vec{j} \circ (\tau + J)]\},\\
C_q &= \Big\{ \Big[ (2i + q + 1) \circ \vec{j} \circ (\tau + J) :\\
&\qquad (2i + q + 1 + \vartheta(d + (s+1)|\vec{j}| - q + 1)) \circ \vec{j} \circ \left(\frac{q}{s} - |\vec{j}|, \tau + J\right) \Big] \Big\}\\
&\quad \cup \Big\{ \Big[ (2i + q) \circ \vec{j} \circ (\tau + J) :\\
&\qquad (2i + q + \varphi(d + (s+1)|\vec{j}| - q)) \circ \vec{j} \circ \left(\frac{q}{s} - |\vec{j}|, \tau + J)\right) \Big] \Big\}.
\end{aligned}$$

There are altogether $(1 + 1/s)(\mathrm{LG}(N') + K) + (s+1)K$ layers in the second part. In this part layers $L_p$ again represent layers of $F_{N'}$ inside the columns (as in $T(\cdot)$). *Correction area* of a column is the set of its registers from which a single displaced 1 is guaranteed to get to or below the border by applying only layers $L_p$. Layers

$C_q$ represent transfers of active 1's which are beyond correction area to columns not containing displaced 1's.

From the way layers $C_q$ are defined we see that only one transfer to a given column can occur during the whole time of computations. Active 1's are transferred from column $\vec{j} \circ (\tau + J)$ to column $\vec{j'} \circ (\tau + J)$ where $|\vec{j'}| = |\vec{j}| + 1$ (since $\vec{j'} = \vec{j} \circ (q/s - |\vec{j}|)$). All transferred 1's after the transfer are in a row at the distance not bigger than $\varphi(d + (s+1)|\vec{j'}| - q - 1)$ from the border. Thus they are in correction area of their new column. At most fraction $1 - 1/f_{s+1}$ of active 1's is transferred remaining active after the transfer. Thus the following fact holds:

FACT 4.4. *A column with the index* $\vec{j} \circ (\tau + J)$ *contains not more than* $t \cdot (1 - 1/f_{s+1})^{|\vec{j}|}$ *active 1's.*

The fact above is the reason why we do not need columns for $|\vec{j}| > K$. Even if they were present, no displaced 1's would get to them. Because all displaced 1's are at the end of computations in rows $b$,$b-1$ or $b-2$ the following fact holds:

FACT 4.5. *The second part of the network reduces the dirty area to at most three rows.*

This way the network $C(\cdot)$ reduces dirty area to at most $3t\binom{J+K}{K} = ct(\log N)^{c_s \log t}$ registers. This proves the following lemma:

LEMMA 4.6. *Network* $C(s, N, t)$ *is a* $(t, ct(\log N)^{c_s \log t})$*-partial-correction unit for some constant* $c_s$ *depending on* $s$. *This network has depth*

$$\alpha \left(1 + \frac{1}{s}\right) \log N + c_s(1 + o(1)) \log t \log \log N$$

**5. Tolerance and Correction Networks.** Now we show how having partial-tolerance and partial-correction networks we can obtain tolerance and correction networks of almost the same depth. The solutions presented in this and the next section are intended to be as simple as possible and are not optimal. Unfortunately the only better constructions the author could think about were much more complicated technically. Describing them would make these sections very boring. The only advantage would be reducing some constants not estimated in this paper.

The problem which is often encountered in construction of comparator networks is sorting inputs with dirty areas of a small size. Assume we can reduce dirty area of $t$-disturbed sequence of 0's and 1's to size $\Delta$. The question is how many layers and comparators a comparator network needs to 'clean' this dirty area i.e. to finish sorting. We have two versions of this question. One if we require fault-tolerance the other if we do not. This question is answered by two easy lemmas:

LEMMA 5.1. *Assume that there exists a* $t$*-tolerance network* $X_N$*, that for input size* $N$ *has depth* $\delta(N, t)$ *and* $\gamma(N, t)$ *comparators. Then there exists a comparator network that sorts any* $x$*-disturbed input with a dirty area of size at most* $\Delta$ *if it has not more than* $t - x$ *faulty comparators. This network has depth* $2\delta(2\Delta, t)$ *and* $\frac{N}{\Delta}\gamma(2\Delta, t)$ *comparators.*

LEMMA 5.2. *Assume that there exists a* $t$*-correction network* $X_N$*, that for input of size* $N$ *has depth* $\delta(N, t)$ *and* $\gamma(N, t)$ *comparators. Then there exists a comparator network that sorts any* $t$*-disturbed input with a dirty area of size at most* $\Delta$*. This network has depth* $2\delta(2\Delta, t)$ *and* $\frac{N}{\Delta}\gamma(2\Delta, t)$ *comparators.*

*Proof of both lemmas.* We index the registers with integers $1, \ldots, N$. The network consists of two parts $\delta(2\Delta, t)$ layers each. The first part consists of networks $X_{2\Delta}$ on each set of registers:

$$S_{2i} = \{r_{2i\Delta+1}, r_{2i\Delta+2}, \ldots, r_{2i\Delta+2\Delta}\}.$$

The second part is are the networks $X_{2\Delta}$ on each set of registers:

$$S_{2i+1} = \{r_{(2i+1)\Delta+1}, r_{(2i+1)\Delta+2}, \ldots, r_{(2i+1)\Delta+2\Delta}\}.$$

This network cleans the dirty area because this area is contained in at least one $S_i$. □

Now having these cleaning networks we can formulate the main result of this section. We are going to prove is that to produce a good $t$-tolerance(-correction) network it is enough to construct $(t, \Delta)$-partial-tolerance(-correction) network $Y_N$ having small depth and reasonably small function $\Delta$ and $t$-tolerance(-correction) network $X_N$ of not too big depth and small number of comparators. In such case we can construct $t$-tolerance(-correction) network of almost the same depth as $Y_N$ and having roughly speaking twice as many comparators as $X_N$ has. We call these reductions refinement lemmas. We formulate and prove them at once for tolerance and correction networks.

LEMMA 5.3 (refinement lemma). *Assume we have a comparator network $Y_N$ which is $(t, \Delta(N,t))$-partial-tolerance(-correction) network of depth $\delta'(N,t)$. We also have a $t$-tolerance(-correction) network $X_N$ of depth $\delta(N,t)$ and having $\gamma(N,t)$ comparators. Then for any $M$ there exists a $t$-tolerance(-correction) network for any input size $N$ of depth*

$$\delta(M,t) + \delta'\left(\frac{Nt}{M}, t\right) + 2\delta\left(\frac{4M\Delta}{t} + 2M, t\right)$$

*with the number of comparators not bigger than*

$$\frac{N}{M}\gamma(M,t) + \frac{Nt}{M}\delta'\left(\frac{Nt}{M}, t\right) + \frac{Nt}{2M\Delta + Mt}\gamma\left(\frac{4M\Delta}{t} + 2M, t\right)$$

*where $\Delta = \Delta(Nt/M, t)$.*

*Proof.* Let register indexes be pairs $(i,j)(i \in \{1, \ldots, N/M\}, j \in \{1, \ldots, M\})$. Our network consists of three main parts.

In the first part we apply $X_M$ in each row separately. This requires $\delta(M,t)$ layers and $\frac{N}{M}\gamma(M,t)$ comparators. The result of this part is that displaced 0's are moved to the first $t$ columns, and displaced 1's are moved to last $t$ columns (except maybe one row at the border between 0's and 1's).

In the second part we use two copies of $Y = Y_{Nt/M}$. The first copy is applied to all registers of last $t$ columns to deal with the displaced 1's. In the same time we deal with first $k$ columns in which there are all displaced 0's. We notice, that $Y$ reversed upside-down can deal with them as ordinary $Y$ does with displaced 1's. So we apply reversed $Y$ to first $t$ columns. All of this requires $\delta'\left(\frac{Nt}{M}, t\right)$ layers and at most $\frac{Nt}{M}\delta'\left(\frac{Nt}{M}, t\right)$ comparators. The result of this part is reduction of dirty area to at most $2\frac{M\Delta}{t} + M$ registers.

The third part is cleaning network (based on $X_N$) for dirty area of size $\frac{2M\Delta}{t} + M$ which requires $2\delta\left(\frac{4M\Delta}{t} + 2M, t\right)$ layers and $\frac{Nt}{2M\Delta+Mt}\gamma\left(\frac{4M\Delta}{t} + 2M, t\right)$ comparators. □

Now we show how we can use refinement lemmas to construct tolerance and correction networks of a small depth. In the construction of $t$-tolerance network we apply the Piotrów's network [5].

THEOREM 5.4. *There exists a constant c such that for an arbitrary s there exists a t-tolerance network of depth:*

$$\alpha\left(1+\frac{1}{s}\right)\log N + c\log\log N + (2s+c)t$$

*having $O(Nt)$ comparators.*

*Proof.* We defined $(t, t^2+t)$-partial-tolerance network $T(s,N,t)$, which has depth $\alpha(1+1/s)\log N + (2s+c')t$. Theorem follows from refinement lemma applied to $X_N$ being Piotrów's network, $Y_N = T(s,N,t), M = t\log N$. ☐

The above network is practical for small $t$. If we fix $t$ and take $s = \sqrt{\log N}$, then we obtain a $t$-tolerance network of depth $\alpha\log N + O(\sqrt{\log N})$.

Similarly as for fault tolerant networks we can now construct a $t$-correction network applying refinement lemma.

THEOREM 5.5. *For any integer s there exists a t-correction network of depth*

$$\alpha\left(1+\frac{1}{s}\right)\log N + c'_s(\log t\log\log N)^2.$$

*for some constant $c'_s$ depending on s.*

*Proof.* We apply refinement lemma taking $Y_N = C(s,N,t), X_N$–Batcher network, $M = t\log N$. ☐

This network has depth $\alpha(1+1/s)\log N + o(\log N)$ for $N \to \infty$ and $t = o\left(2^{\sqrt{\log N}/\log\log N}\right)$. We know, that $c'_s = O(c_s^2) = O(f_s^2) = O(2^{2s})$. We can take $s = \log\log\log N$ and obtain the following corollary:

COROLLARY 5.6. *For any t there exists a t-correction network of depth*

$$\alpha\left(1+\frac{1}{\log\log\log N}\right)\log N + c\log^2 t\log\log^4 N \sim \alpha\log N$$

*for some constant c.*

We can also apply refinement lemma once again taking the network from the last theorem for $s=1$ as $X_N$. We put $Y_N = C(s,N,t), M = t\log N$ and get the following corollary.

COROLLARY 5.7. *For any integers s, t there exists a t-correction network of depth*

$$\alpha\left(1+\frac{1}{s}\right)\log N + c''_s\log t\log\log N + o(\log\log N).$$

*for some constant $c''_s$ depending on s.*

Unfortunately the construction from the corollary works only for relatively small $t$.

**6. Minimizing the Number of Comparators.** First we should know what the minimal possible numbers of comparators for $t$-tolerance and $t$-correction networks are. Any $t$-tolerance network has at least $t$ comparators going up from any register different from the highest one to the register with the index higher by one (to make it impossible for all of them to be faulty for 1-disturbed input). So it has at least $(N-1)t = \Omega(Nt)$ comparators. Any $t$-correction network has to be a $t$-selector which forces it to have $\Omega(N\log t)$ comparators [7]. These asymptotic lower bounds on the numbers of comparators in correction networks prove to be achieved.

A $t$-tolerance network having asymptotically optimal number of comparators (i.e. $O(Nt)$) is one from the previous section. It has depth $\alpha(1+1/s)\log N+O(\log\log N+st)$.

An optimal $t$-correction network is constructed using refinement lemma. Similar techniques to those we use in this section can be applied to reduce numbers of comparators of practical correction networks. The simplest way to make these practical constructions is to use Batcher network instead of AKS in what follows. Unfortunately we were not able to find a $t$-correction network with asymptotically optimal number of comparators without using AKS network, so our further constructions are not practical. First we construct a network that is asymptotically optimal in the sense of the number of comparators but is not if the depth is concerned.

LEMMA 6.1. *There exists a $t$-correction network that for some constant $c$ and any input size $N$ has depth $cN\log t/t$ and at most $cN\log t$ comparators.*

*Proof.* Let AKS denote a sorting network which has depth $\frac{c}{2}\log t$ for input of size $2t$ [1]. It has at most $\frac{c}{2}t\log t$ comparators. We index registers with integers $1,\ldots,N$. Let $S_i$ be the following set of registers

$$S_i=\{r_{(i-1)t+1},r_{(i-1)t+2},\ldots,r_{(i-1)t+2t}\}.$$

Our network consists of $2N/t-3$ parts $c\log t$ layers each. Each part consists of a single AKS network on register set $S_i$. Thus we apply AKS subsequently to

$$S_1,S_2,\ldots,S_{(N/t)-1},S_{(N/t)-2},\ldots,S_1.$$

It is easy to see that what we constructed is really a $t$-correction network. □

When we have the $t$-correction network from the last lemma we can put it as $X_N$ to refinement lemma taking $M=\frac{t\log N}{\log t}$. As $Y_N$ we can use AKS which is a $(t,0)$-partial-correction network. This way we obtain the following corollary:

COROLLARY 6.2. *There exists a $t$-correction network of depth $O(\log N)$ having $O(N\log t)$ comparators.*

Further on we can take correction network from the last lemma as $X_N$, $C(s,N,t)$ as $Y_N$ and $M=t\log N$. As a result by refinement lemma we obtain the following corollary:

COROLLARY 6.3. *For any integer $s$ there exists a $t$-correction network of depth*

$$\alpha(1+1/s)\log N+c_s'\log t\log\log N$$

*for some constant $c_s'$ depending on $s$ which has $O(N\log t)$ comparators.*

A natural question whether applying AKS sorting network is necessary in above constructions of correction networks can arise. We can justify the use of AKS by the following proof of lower bound for the number of comparators in correction network. Let us have a $t$-correction network for $N$ entries with the minimum possible number of comparators. We can notice, that any $t$ subsequent registers have to form a sorting network together with comparators that connect them. The lower bound for the number of comparators in a sorting network for $t$ entries is $\Omega(t\log t)$. Dividing all registers into groups of $t$ subsequent registers we get the lower bound of $\Omega(N\log t)$ comparators in $t$-correction network. If we had a correction network not based on AKS and having $O(N\log t)$ comparators, then some groups of $t$ subsequent registers would be connected with $O(t\log t)$ comparators. These $O(t\log t)$ comparators would form a sorting network different from AKS. Unfortunately we do not know any different from AKS sorting networks with $O(t\log t)$ comparators.

**7. Conclusions.** We constructed $t$-tolerance and $t$-correction networks of depths $\sim \alpha \log N$ for fixed $t$. This is less than depth of 1-correction network found by Schimmler and Starke [5]. Network $T(\cdot)$ seems to be better for practical purposes although it is worse than $C(\cdot)$ for combinations of $N$ and $t$ where $N$ is big and $t \geq \log N$. Some considerations we did not include in this paper seem to indicate that the following conjecture is true. This conjecture was originally posed by Mirek Kutyłowski – authors only contribution is the constant $\alpha$.

CONJECTURE 7.1. *The lower bound for depth of 1-correction network is*

$$\alpha \log N - c.$$

*for some small constant $c$.* Because the author was unable to find 2-correction networks having depth asymptotically better than $T(\cdot)$, he dares to pose another conjecture concerning 2-correction networks.

CONJECTURE 7.2. *The lower bound for depth of 2-correction network is*

$$\alpha \log N + c\sqrt{\log N}$$

*for some constant $c > 0$.*

**Acknowledgments.** Author wishes to thank Mirek Kutyłowski, Krzysiek Loryś and Marek Piotrów for presenting the problems, helpful discussions and their encouragement to write this paper.

REFERENCES

[1] M. Ajtai, J. Komolós, E. Szemerédi, Sorting in $c \log n$ parallel steps, *Combinatorica* 3 (1983), 1-19.

[2] K.E. Batcher, Sorting networks and their applications, in *AFIPS Conf. Proc.* 32 (1968), 307-314.

[3] M. Kik, M. Kutyłowski, M. Piotrów, Correction Networks, in *Proc. of 1999 ICPP*, 40-47.

[4] M. Piotrów, Depth Optimal Sorting Networks Resistant to $k$ Passive Faults in *Proc. 7th SIAM Symposium on Discrete Algorithms* (1996), 242-251 (also accepted for *SIAM J. Comput.*).

[5] M. Schimmler, C. Starke, A Correction Network for $N$-Sorters, *SIAM J. Comput.* 18 (1989), 1179-1197.

[6] G. Stachowiak, Fibonacci Correction Networks, in *Algorithm Theory – SWAT 2000* , M Halldórsson (Ed.) , LNCS 1851, Springer 2000, 535-548.

[7] A.C. Yao, Bounds on Selection Networks, *SIAM J. Comput.* 9 (1980), 566-582.

[8] A.C. Yao, F.F. Yao, On Fault-Tolerant Networks for Sorting, *SIAM J. Comput.* 14 (1985), 120-128.