Periodic Correction Networks ⋆

Marcin Kik

Institute of Computer Science, Wrocław University
kik@tcs.uni.wroc.pl

Abstract. We present a comparator network that sorts sequences obtained by changing a limited number of keys in a sorted sequence. The network that we present is periodic and has a depth 8. The time required by this algorithm is $O(\log n+k)$ with a small constant hidden by "Oh" notation (n is the total number of keys, k is the number of changed keys).

1 Introduction

Sorting on comparator networks. Sorting is one of the fundamental computer science problems both in theory and practice. One of the models for designing sorting algorithms are comparator networks. We are given a set of registers $\{R_0, R_1, \ldots, R_{n-1}\}$, each register capable of holding a single key. The goal is to sort these keys, i.e. to relocate them so that their ordering agrees with the ordering of registers. For this purpose, we apply *compare-exchange operations*. Such an operation compares numbers x, y stored in R_i and R_j and stores $\min\{x, y\}$ in R_i and $\max\{x, y\}$ in R_j. In the situation described, we say also that there is a comparator (R_i, R_j) between R_i and R_j that performs the compare-exchange operation. Comparators might be executed in parallel provided that no register is involved in more than one operation at a time. Such a collection of comparators is called a *layer*. *Comparator network* is an algorithm which executes a fixed set of comparators (independent of data). The comparators of a comparator network are organized into several layers; the number of these layers is called *depth* of the network and corresponds to the execution time. The input keys are placed in the registers of the network, the ith key in register R_i. Once a sorting algorithm terminates its execution, the keys stored in $R_0, R_1, \ldots, R_{n-1}$ form a nondecreasing sequence.

There are comparator networks that sort n keys in $O(\log n)$ parallel steps [1] (with $\log n$ being an obvious lower bound for the number of parallel steps). However, they are not practical because of a big constant hidden by the big "Oh" notation.

Correcting disturbed sequences. One of the problems arising in practical computing is re-sorting data that once has been sorted, but a limited number of keys has been changed. If the number of such keys is k, we say that the sequence is *k-disturbed*. Of course, for sorting k-disturbed sequences one may apply general sorting algorithms, but this might be less efficient than using specially tailored methods.

⋆ partially supported by Komitet Badań Naukowych, grant 8 T11C 032 15

The problem considered in this paper is how to sort efficiently k-disturbed sequences on comparator networks for k substantially smaller than n. We call such networks *k-correction networks*. Some partial solutions to this problem are known: Schimmler and Starke [8] present a network with $O(n)$ comparators correcting 1-disturbed sequences in time $2\log n - 1$. A network of depth $4\log n + O(\log^2 k \log\log n)$ that sorts k-disturbed sequences for $k \leq n$ is presented in [4]. Some further results have been reported by G. Stachowiak.

Periodic comparator networks. A *periodic network* processes the input sequence in many iterations, i.e. the output of the network is processed in the next iteration as the input. Thus, the network may have even a constant depth and still be capable of sorting arbitrary sequences. A motivation for periodic comparator networks is to reduce amount of hardware necessary to implement a sorting algorithm.

For the sorting problem, many periodic comparator networks have been designed. The first significant step was a periodic network of depth $\log n$ sorting in $\log n$ iterations [3]. Later, quite fast networks of a constant depth have been invented for many architectures ([9], [5], [6]).

In this paper we present periodic comparator network of a constant depth capable of sorting disturbed sequences:

Theorem 1. *There is a periodic comparator network of depth* 8 *that sorts* k*-disturbed sequences in time* $O(\log n + k)$.

2 Preliminaries

One of the fundamental observations concerning comparator networks is that a comparator network N sorts all sequences if and only if N sorts all zero-one sequences (i.e. sequences of zeroes and ones) ([7], page 224). This principle let us restrict the analysis of a network to zero-one inputs only. The same phenomenon holds for k-disturbed sequences, in order to prove that a network sorts all k-disturbed sequences it suffices to show it for sorted zero-one sequences in which at most k keys have changed the value from zero to one or from one to zero.

For a zero-one sequence with x zeroes the *zeroes area* (respectively, *ones area*) is the set of registers R_0 through R_{x-1} (respectively, R_x through R_{n-1}). A *displaced* key is a one in the zeros area or a zero in the ones area. It is easy to show (see [4]) that any k-disturbed zero-one sequence contains at most k displaced zeroes and at most k displaced ones.

The simplest periodic sorting network is *odd-even transposition network*. Its first layer contains comparators (R_i, R_{i+1}), for even i, the second layer consists of comparators (R_i, R_{i+1}), where i is odd. This is an extremely simple architecture, however the sorting time is n.

A common strategy in analysis of a comparator network is to isolate a *dirty area*, that is a set of registers $R_j, \ldots, R_h$, such that $R_0, \ldots R_{j-1}$ contain only zeroes and $R_{h+1}, \ldots R_{n-1}$ contain only ones. Then the *clean* areas remain clean and we may restrict our attention to comparators with both endpoints in the dirty area.

It is relatively easy to design a periodic 1-correction network with runtime $O(\log n)$. The problem that we have to solve is that displaced elements may block each other while going into proper positions. This phenomenon known for arbitrary correction networks becomes a nasty one for periodic correction networks. Due to the periodic nature of the algorithm we cannot adjust the steps in order to respond appropriately to problems arising at different phases of sorting.

3 Periodic k-correction Network

Definition of the Network. In this section we present the network $P_{l,w'}$ ($l > 1$ and $w' > 1$ are parameters of the construction) with the properties stated in Theorem 1, for $w' = 6k$. Let us remark at this point that some features of $P_{l,w'}$ are introduced just to enable the analysis, and may be unnecessary for a good performance in practice.

Let $w = 2(l + 1 + w')$, $h = 2^l$. The input size of $P_{l,w'}$ is $n = wh$. Let $R = \{R_0, \ldots, R_{n-1}\}$ be a set of registers. We define a function $r : \{0, \ldots, w-1\} \times \{0, \ldots, h-1\} \to R$ by $r(x, y) = R_{x+wy}$. We arrange the registers of $P_{l,w'}$ in a matrix, where the register $r(x, y)$ is placed in column x and row y. (The row numbers increase downwards, i.e. row y is above the row $y + 1$.)

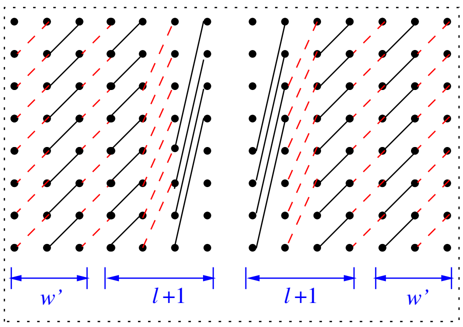


Fig. 1. Back-jump layers M_1 (solid lines) and M_3 (dashed lines).

The network $P_{l,w'}$ has 8 layers $(M_0, M', M_1, M', M_2, M', M_3, M')$. First we define M_1 and M_3 (called *back-jump* layers): Let $C_x = \{(r(x + 1 + \frac{w}{2}, y), r(x + \frac{w}{2}, y + 2^{\max\{0,l-1-x\}})) \in R^2\}$. Let C'_x be the set of comparators (R_{n-1-t}, R_{n-1-s}) such that (R_s, R_t) is in C_x. M_1 (respectively, M_3) is the union of the sets $C_x \cup C'_x$, for even (respectively, odd) x, $0 \le x < \frac{w}{2} - 1$ (see Fig. 1).

M_0 and M_2 (called *horizontal* layers) are defined as follows (see Fig. 2 (a)). Let $D_x = \{(R_s, R_{s+1}) \in R^2 \mid s \bmod w = x\}$. Then M_0 (respectively, M_2) is the union of D_x, for $0 \le x \le w - 1$, where $x - \frac{w}{2}$ is even (respectively, odd) and $x \ne \frac{w}{2} - 1$.

We also define a *left-right* layer M' as $\{(r(x, y), r(w - 1 - x, y)) \mid 0 \le x < \frac{w}{2}\}$ (see Fig. 2 (b)).

We partition the set of registers R into the *left set* $S_0 = \{r(x, y) \mid 0 \le x < \frac{w}{2}\}$ and the *right set* $S_1 = R \setminus S_0$. The members of S_0 (respectively, S_1) are called

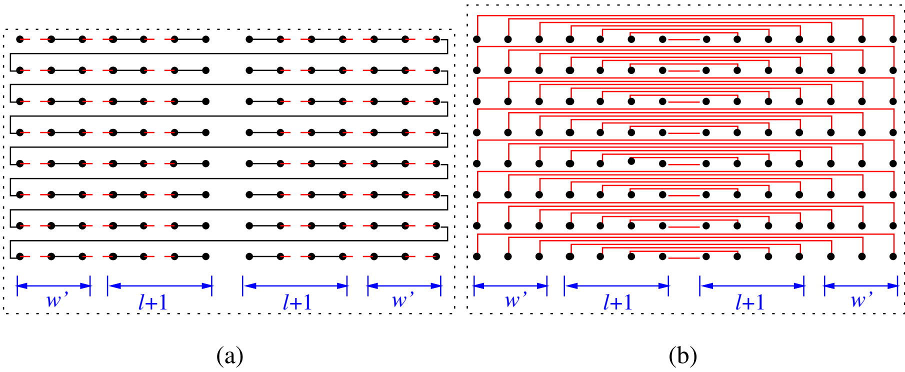


Fig. 2. Layers M_0 (solid lines) and M_2 (dashed lines) (a), and M' (b).

left (respectively, *right*) registers. Fig. 3(b) presents $P_{3,3}$ in a *folded* state (i.e. with S_0 rotated 180 degrees around the central vertical axis). For any register $r = r(x, y)$, we define a *shadow* of r, denoted by $\mathrm{s}(r)$, as $r(w-1-x, y)$. For $X \subseteq R$, $\mathrm{s}(X) = \{\mathrm{s}(r) \mid r \in X\}$. Note that in the folded state, each register is in the same place as its shadow, and that M' contains comparators of the form $(\mathrm{s}(r), r)$, for $r \in S_1$.

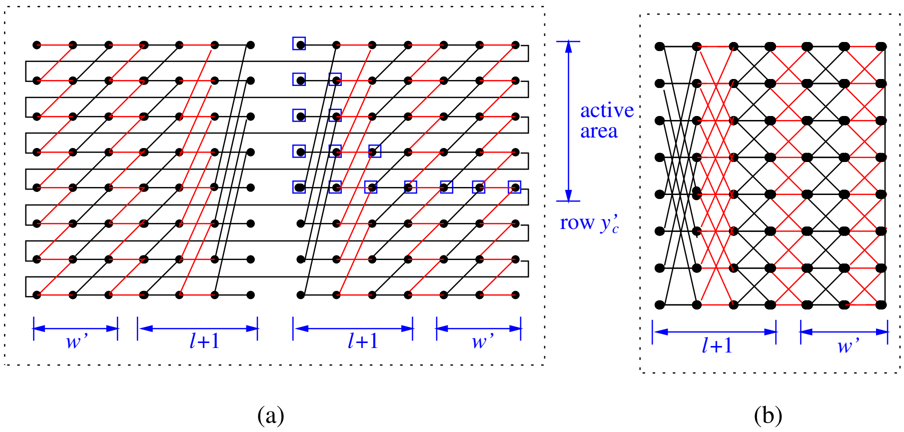


Fig. 3. $P_{3,3}$ without the left-right comparators (a), and folded $P_{3,3}$ (b).

Sketch of the Runtime Analysis of $P_{l,w'}$. Let $L = (L_0, \ldots, L_7)$ be the sequence of layers of $P_{l,w'}$ (i.e. $L_{2i+1} = M'$ and $L_{2i} = M_i$). Let c be an arbitrary zero-one k-disturbed sequence. We show that $P_{l,w'}$ sorts c in $O(w' + l) = O(k + \log n)$ iterations. Let y_c' be the index of the first row of registers that intersects the ones area. For $t \geq 0$, let c_t be a sequence obtained after execution of t steps (i.e layers) of the iterated $P_{l,w'}$

on c. We use $c(r)$ to denote the r-th element of the sequence c (i.e. the value stored in register R_r). We apply a technique that is reverse to zero-one principle in some sense. Let c' be a sequence defined as follows.

$$c'(r) = \begin{cases} 0 & \text{if } c(r) = 0, \\ \sum_{p=0}^{r} c(p) & \text{if } c(r) = 1 \text{ and } R_r \text{ is above row } y'_c, \\ k+1 & \text{if } c(r) = 1 \text{ and } R_r \text{ is below row } y'_c - 1. \end{cases}$$

The sequence c has at most k displaced ones. Thus the registers above the row y'_c, which contain displaced ones in c, contain the values from the range $\{1, \ldots, k\}$ in c', every value occurring exactly once. The sequences c'_t and c''_t are defined as follows:

- $c''_0 = c'$.
- For $t \geq 0$, c'_t is the sequence c''_t with all the values from the range $\{1, \ldots, k\}$ below the row $y'_c - 1$ replaced by $k + 1$.
- For $t \geq 0$, c''_{t+1} is the output of $L_{t \bmod 8}$ applied to c'_t.

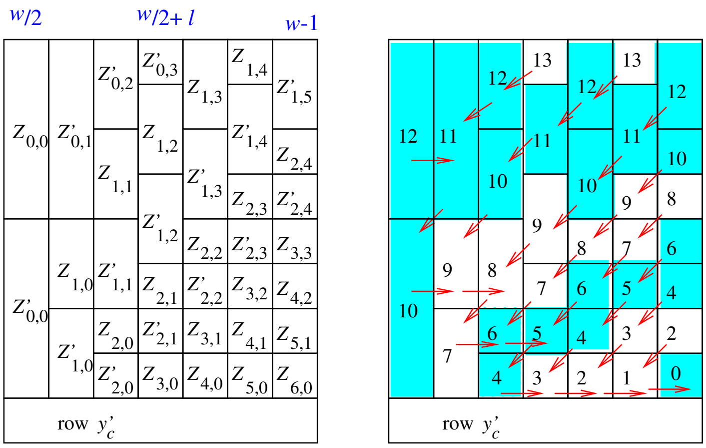
Note that $c_t(r) = 1$ if and only if $c'_t(r) > 0$. We will show that for $t = O(l + w')$ there are only zeroes above the row $y'_c - 1$ in c'_t (and hence in c_t).

Active area and stoppers. Below, we start to investigate in detail the fine structure of $P_{l,w'}$. We call the set of registers above the row y'_c an *active area* and denote it by S' (see Fig. 3 (a)). The registers of S' are called *active*. Let $S'_0 = S_0 \cap S'$ and $S'_1 = S_1 \cap S'$. We call $r \in S'_1$ a *stopper* if and only if there is no back-jump comparator of the form (r, r') such that r' is in the active area. The stoppers on Fig. 3 are marked with the boxes. By *active values* we mean the values from the range $\{1, \ldots, k\}$ stored in the active registers. Note that any active value may disappear (be replaced by $k + 1$) if it is compared with a zero from outside S'.

Zones and the levels of registers. We partition the set S'_1 into zones $Z_{i,j}$ and $Z'_{i,j}$ defined as follows (see Fig. 4 (a)):

- for $0 \leq x \leq l - 1$, $Z_{x,0}$ (respectively, $Z'_{x,0}$) is the set of all stoppers $r(\frac{w}{2} + x, y)$ such that $y'_c - y > 2^{l-1-x}$ (respectively, $y'_c - y \leq 2^{l-1-x}$).
- for $l \leq x \leq \frac{w}{2} - 1$, $Z_{x,0} = \{r(\frac{w}{2} + x, y'_c - 1)\}$,
- for $x' > 0$, $Z_{x,x'}$ (respectively, $Z'_{x,x'}$) is the set of $r \in S'_1$ such that there is a back-jump comparator (r, r') with $r' \in Z_{x,x'-1}$ (respectively, $r' \in Z'_{x,x'-1}$).

It is easier to analyze the movements of active values between the zones than their detailed routes. A *zones tree* T is a directed graph with the nonempty zones as vertices and the set of arcs $E = E_1 \cup E_2$ (see arrows on Fig. 4 (b)), where E_1 contains the arcs of the form $(Z_{x,x'+1}, Z_{x,x'})$ and $(Z'_{x,x'+1}, Z'_{x,x'})$ (*back-jump arcs*) and $E_2 = E_{2,1} \cup E_{2,2} \cup E_{2,3}$ (*horizontal arcs*), where $E_{2,1} = \{(Z_{x,0}, Z'_{x,1}) \mid 0 \leq x \leq l - 1\}$, and $E_{2,2} = \{(Z'_{x,0}, Z_{x+1,0}) \mid 0 \leq x \leq l - 1\}$, and $E_{2,3} = \{(Z_{x,0}, Z_{x+1,0}) \mid l \leq x \leq \frac{w}{2}\}$. The *root* of T is $Z_{\frac{w}{2}-1,0}$. For each zone Z we define its *level* (denoted by $\mathrm{l}(Z)$) as its distance from the root of T, and, for each $r \in Z \cup \mathrm{s}(Z)$, $\mathrm{l}(r) = \mathrm{l}(Z)$. For each active value i, let *level of* i in c'_t is the level of active register that contains i or zero if i does not exist in c'_t. The levels of the zones are displayed on Fig. 4 (b). Let l'_T denote the maximal level of a zone in T. By considering the path from arbitrary vertex of T to the root, it is easy to note that l'_T is $O(w' + l)$.



(a) (b)

Fig. 4. Partition of the right active registers into the zones (a) and the levels of the zones (b). The arrows on (b) represent the arcs of the tree of zones T.

Releasing the registers from the active values. For each $t \geq 0$, for active i, we say that a register R_r *is released* from i at step t if and only if for each $t' \geq t$, $c'_{t'}(r) \neq i$.

Consider the following simple example: Suppose we have the odd-even transposition network with an input consisting of the k positive values: $1, \ldots, k$, placed in arbitrary registers and zeroes placed in all the remaining registers. Note that after the first computation step, the register R_0 is released from k. After the second step the registers R_0 and R_1 are released from k and thus after the third step the register R_0 is released from $k-1$. In such a way we can define, for each $t \geq 0$ and $i > 0$, the set of registers that are released from i at step t. Note that the border of the area released from $i-1$ is adjacent to the border of the area released from i.

We apply analogous reasoning. For each $l \geq 0$, we define A_l as $\{r \in S'_1 \mid \mathrm{l}(r) \leq l\}$. Note that $A_{l'_T} = S'_1$ and $A_0 = Z_{\frac{w}{2}-1,0}$. We partition the set of levels into *groups of levels*. Note that for $0 < x < l$, we have $\mathrm{l}(Z'_{x-1,0}) - \mathrm{l}(Z'_{x,0}) = 3$. Let $b = \mathrm{l}(Z'_{0,0}) \bmod 3$. We define the ith *group of levels* as $\mathcal{G}_i = \{j \mid 3i + b \leq j \leq 3i + b + 2\}$. The zones with the levels in the odd and even groups have been depicted by different shades on Fig. 4 (b). By the definition of b, the level of each zone $Z'_{x,0}$ is a minimum of some group of levels. The *first phase* of the computation consists of $6l'_T$ iterations.

Lemma 1. *For each active value i, $S' \setminus (A_{\max \mathcal{G}_{2(k-i)}} \cup \mathrm{s}(A_{\min \mathcal{G}_{2(k-i)}}))$ is released from i after $6l'_T$ iterations.*

Sketch of the proof. The proof is based on the following claims.

Claim. For each active value i, if $S' \setminus (A_{\max \mathcal{G}_j} \cup \mathrm{s}(A_{\min \mathcal{G}_j}))$ is released from the values greater than i at step t, and i is inside $A_{\min \mathcal{G}_{j+1}} \cup \mathrm{s}(A_{\max \mathcal{G}_j})$ in sequence c'_t, then $S' \setminus (A_{\max \mathcal{G}_{j+1}} \cup \mathrm{s}(A_{\min \mathcal{G}_{j+1}}))$ is released from i at step t.

There is no greater value than i in $S' \setminus (A_{\max \mathcal{G}_j} \cup \mathrm{s}(A_{\min \mathcal{G}_j}))$. Thus i can leave $\mathrm{s}(A_{\max \mathcal{G}_j})$ by a left-right comparator or being pulled back by a greater value directly to $A_{\max \mathcal{G}_j}$, or by going forward through a horizontal or back-jump comparator to a shadow of a zone with level $\min \mathcal{G}_{j+1}$. In the last case i is moved by the next left-right layer to $A_{\min \mathcal{G}_{j+1}}$. i can leave $A_{\min \mathcal{G}_{j+1}}$ by entering some zone with level $\min \mathcal{G}_{j+1} + 1$ or by going from the column $w-1$ to 0. In the first case i is moved by the next back-jump layer back to $A_{\min \mathcal{G}_{j+1}}$. In the second case i either enters $\mathrm{s}(A_{\max \mathcal{G}_j})$ or is moved by the next left-right layer back to $A_{\min \mathcal{G}_{j+1}}$.

Claim. For each active value i, if $S' \setminus (A_{\max \mathcal{G}_j} \cup \mathrm{s}(A_{\min \mathcal{G}_j}))$ is released from the values greater than i after iteration t (i.e. at step $8t$), and $S' \setminus (A_{\max \mathcal{G}_{j+2}} \cup \mathrm{s}(A_{\min \mathcal{G}_{j+2}}))$ is released from i after iteration t, then within the next 6 iterations i must visit some register from $A_{\min \mathcal{G}_{j+1}} \cup \mathrm{s}(A_{\max \mathcal{G}_j})$ thus releasing $S' \setminus (A_{\max \mathcal{G}_{j+1}} \cup \mathrm{s}(A_{\min \mathcal{G}_{j+1}}))$.

We skip the details of the proof. There are no greater values than i in $S' \setminus (A_{\max \mathcal{G}_j} \cup \mathrm{s}(A_{\min \mathcal{G}_j}))$. Thus while i is outside $A_{\min \mathcal{G}_{j+1}} \cup \mathrm{s}(A_{\max \mathcal{G}_j})$ it must enter $A_{\max \mathcal{G}_{j+2}}$ by a left-right comparator, and then:

- move through the back-jump comparators until it reaches a stopper,
- move through the horizontal and back-jump comparators according to the arcs of T (or by a horizontal comparator directly from a lower half of some $Z'_{x,0}$ to $Z'_{x+1,0}$).

During each of the above steps, level of i is decreased by at least 1 and each iteration contains at least one such step.

By the last claim we have: for $t \geq 0$, for each active i, $S' \setminus (A_{l_{t,i}} \cup \mathrm{s}(A_{l_{t,i}-2}))$ is released from i after the iteration $6 \cdot t$, where $l_{t,i} = \max \mathcal{G}_{\max\{2(k-i), l'_T - t + 2(k-i)\}}$.

Thus after the first phase all active values will have the levels not greater than $\max \mathcal{G}_{2(k-1)} = 6k - 4 + b$. Since $b \leq 2$ and $w' = 6k$, they will remain in $A_{w'-1} \cup \mathrm{s}(A_{w'-1})$. This part of a network has a very simple structure. It can be shown that after $O(w')$ iterations of the next (*second*) phase $S' \setminus (A_{k-i} \cup \mathrm{s}(A_{k-i}))$ is released from i, for each active value i. We add one more iteration to the second phase to ensure that both k and $k-1$ are moved to A_0 and A_1 respectively.

Final smoothing of displaced elements. After the second phase each active i is inside $A = A_{k-i} \cup \mathrm{s}(A_{k-i})$. The final goal is to move them into the last row of A_k, call it B, and its shadow, B'. Now we change our conventions: every nonzero element below active area and in B is replaced by $k+1$ as soon as it arrives there.

To each register r in $A \setminus B$ we assign a label $q_t(r)$ that increases during the computation so that $k - q_t(r)$ is an upper bound on the active value that can still appear in r. Initially, B' is released from k and $k-1$ so initial labels (i.e. q_0) of its registers are 1.5. The registers in $A \setminus (B \cup B')$ have initial labels equal to their levels. For $r \in A \setminus B$, if there is a comparator inside $(A_k \setminus B) \cup B'$, of the form (r, r') or $(\mathrm{s}(r), r')$ such that $q_t(r) - q_t(r') = 0.5$, then all the registers with the label $q_t(r)$ are connected in a similar way to some register with the label $q_t(r) - 0.5$ and we set $q_{t+1}(r) = q_t(r) + 0.5$. (Note that either $q_t(r)$ is an integer and the register r' is released from $k - q(r) + 1$ and $k - q_t(r)$ can move from r to r' in at most single iteration, or we can increase $q_t(r)$ by 0.5 without destroying the bound on the active values that can be in r.) If there is no such comparator, then $q_{t+1}(r) = q_t(r)$.

It is straightforward to show that all registers in $A \setminus (B \cup B')$ have the label q_{2k} greater than k. Hence, after $2k$ iterations of the third phase all the active values are in the row $y'_c - 1$.

We have shown that $P_{l,w'}$ moves all displaced ones below the row $y'_c - 2$ and (by symmetry) all displayed zeroes above the row $y'_c + 2$ in at most $O(l + w')$ iterations. The rows y'_{c-1}, y'_c and y'_{c+1} are then sorted in $O(w)$ iterations, by the comparators of the odd-even transposition network contained in the horizontal and left-right layers.

4 Conclusions

In this paper, we have shown that k-disturbed sequences may be corrected efficiently by constant depth periodic networks. However, our solution remains efficient for small k's only (i.e. $k = o(\log^3 n)$), since for $k = \Omega(\log^3 n)$ the "periodification" [6] of a Batcher sorting network [2] is more efficient than our solution. It is an open question how to obtain correction networks with runtime $c \log n + o(k)$.

Acknowledgment. Thanks to Grzegorz Stachowiak for helpful discussions and to Krzysztof Loryś for comments on this paper.

References

1. M. Ajtai, J. Komlós and E. Szemerédi. Sorting in $c \log n$ parallel steps. Combinatorica, Vol. 3, pages 1–19, 1983.
2. K. E. Batcher. Sorting networks and their applications. Proceedings of 32nd AFIPS, pages 307–314, 1968.
3. M. Dowd, Y. Perl, M. Saks, and L. Rudolph. The periodic balanced sorting network. Journal of the ACM, Vol. 36, pages 738–757, 1989.
4. M. Kik, M. Kutyłowski and M. Piotrów. Correction networks. Proceedings of the IEEE-ICPP, pages 40-47, 1999.
5. M. Kik, M. Kutyłowski and G. Stachowiak. Periodic constant depth sorting network. Proceedings of the 11th STACS, pages 201–212, 1994.
6. M. Kutyłowski, K. Loryś, B. Oesterdiekhoff, and R. Wanka. Fast and feasible periodic sorting networks. Proceedings of the 55th IEEE-FOCS, 1994. Full version to appear in the Journal of ACM.
7. D. E. Knuth. The art of Computer Programming. Volume 3: Sorting and Searching. Addison-Wesley, 1973.
8. M. Schimmler, C. Starke. A Correction Network for N-Sorters. SIAM Journal on Computing, Vol. 6, No. 6, 1989.
9. U. Schwiegelshohn. A short-periodic two-dimensional systolic sorting algorithm. IEEE International Conference on Systolic Arrays, pages 257–264, 1988.