Fast Periodic Correction Networks

Grzegorz Stachowiak

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

Abstract. We consider the problem of sorting \( N \)-element inputs differing from already sorted sequences on \( t \) entries. To perform this task we construct a comparator network that is applied periodically. The two constructions for this problem made by previous authors required \( O(\log n + t) \) iterations of the network. Our construction requires \( O(\log n + (\log \log N)^2 (\log t)^3) \) iterations which makes it faster for \( t \gg \log N \).

Keywords: sorting network, comparator, periodic sorting network.

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 hardware implementations. They can be also implemented as sorting algorithms for parallel computers.

In our approach sorted elements are stored in registers \( r_1, r_2, \ldots, r_N \). Registers are indexed with integers or elements of other linearly ordered sets. A comparator \([i : j]\) is a simple device connecting registers \( r_i \) and \( r_j \) \((i < j)\). It compares the keys they contain and if the key in \( r_i \) is bigger, it swaps the keys. 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 indices by 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 layers which are sets of comparators connecting disjoint pairs of registers. The total time needed by such a network to sort a sequence is proportional to the number of layers called the network’s depth.

Much research concerning sorting networks was done in the past. Most famous results are asymptotically optimal AKS [1] sorting network of depth \( O(\log N) \) and more ‘practical’ Batcher [2] network of depth \( \sim \frac{1}{2} \log^2 N \) (from now on all the logarithms are binary).

Some research was devoted to problems concerning periodic sorting networks. Such a comparator network is applied not once but many times in a series of iterations. The input of the first iteration is the sequence to be sorted. The input of \((i + 1)\)st iteration is the output of \( i \)th iteration. The output of the last iteration should always be sorted. The total time needed to sort an input sequence is the product of the number of iterations...
and the depth of the network. Constructing such networks especially of small constant depth gives hope to reduce the amount of hardware needed to build sorting comparator networks. It can be done by applying the same small chip many times to sort an input. We can also view such a network as a building block of a sorting network in which layers are repeated periodically. Main results concerning periodic sorting networks are presented in the table:

<table>
<thead>
<tr>
<th>Source</th>
<th>Depth</th>
<th># Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPS [3]</td>
<td>log N</td>
<td>log N</td>
</tr>
<tr>
<td>Schwiegelsohn [15]</td>
<td>8</td>
<td>$O(\sqrt{N} \log N)$</td>
</tr>
<tr>
<td>KKS [5]</td>
<td>$O(k)$</td>
<td>$O(N^{1/k})$</td>
</tr>
<tr>
<td>Loryś et al. [9]</td>
<td>3-5</td>
<td>$O(\log^2 N)$</td>
</tr>
</tbody>
</table>

Last row of this table requires some words of explanation. The paper [9] describes a network of depth 5, but a later paper [10] reduces this value to 3. The number of iterations $O(\log^2 N)$ is achieved by periodification of AKS sorting network for which the constant hidden behind big $O$ is very big. Periodification of Batcher network requires less iterations for practical sizes of the input, though it requires the time $O(\log^3 N)$ asymptotically. It is not difficult to show that 3 is the minimal depth of a periodic sorting network which requires $o(N)$ iterations to sort an arbitrary input.

A sequence obtained from a sorted one by $t$ changes being either swaps between pairs of elements or changes on single positions we call $t$-disturbed. We define $t$-correction network to be a specialized network sorting $t$-disturbed inputs. Such networks were designed to obtain a sorted sequence from an output produced by a sorting network having $t$ faulty comparators [13,11,16]. There are also other potential applications in which we have to deal with sequences that differ not much from a sorted one. Let us consider 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. It can be more effective to use a specialized correction unit in such a case, than to apply a sorting network. Results concerning such correction networks are presented in [4,16].

There was some interest in constructing periodic comparator networks of a constant depth, that sort $t$-disturbed inputs. The reason is that the fastest known constant depth periodic sorting networks have running time $O(\log^2 N)$. On the other hand in some applications faster correction networks can replace sorting networks. Two periodic correction networks were already constructed by Kik and Piotrów [6,12]. The first of them has depth 8 and the other has depth 6. Both of them require $O(\log N + t)$ iterations for considered inputs where $N$ is input size and $t$ is the number of modifications. The running time is $O(\log N)$ for $t = O(\log N)$ and the constants hidden behind the big $O$ are small. Unfortunately it is not known how fast these networks complete sorting if $t \gg \log N$.

In this paper we construct a periodic $t$-correction network to deal with $t : \log N \ll t \ll N$. The reason we assume that $t$ is small in comparison to $N$ is the following. If $t$ is about the same as $N$, then the periodification scheme gives a practical periodic sorting network of depth 3 requiring $O(\log^3 N) = O(\log^3 t)$ iterations. Actually we do not hope to get better performance in such a case. Our network has depth 3 and running time: $O(\log N + (\log \log N)^2 (\log t)^3)$. We should mention that in our construction