AEA
Sorting networks

Ghiga, Claudiu-Alexandru
claudiu.ghiga@info.uaic.ro

Nistor, Serban
serban.nistor@info.uaic.ro

November 2nd, 2020

1 Problem description

Sorting networks are data-oblivious algorithms that are used to sort numbers. In other words, regardless of the form that the numerical input data takes, a sorting network will produce as output a monotonically increasing sequence of the provided data. Sorting networks are composed of wires and comparators.

A wire transmits a value from place to place, by either connecting the input of the network to the input of a comparator, the output of a comparator to the input of another comparator or the output of a comparator to the output of the network. If a sorting network contains n wires, than it can sort an arbitrary sequence of n elements $\langle a_1, a_2, \ldots, a_n \rangle$. A comparator is a device that has two inputs x and y and two outputs x' and y' which are assigned the follwing way:

$$x' = \min(x, y)$$
$$y' = \max(x, y)$$

A structure comprised of wires and comparators that does not sort all of its possible inputs is called a comparator network.

An important concept that will be of use later is the zero-one principle, which states that if a sorting network with n wires correctly sorts all the 2^n possible arragements of the set $0, 1$, then it can sort correctly any arbitrary input of n numbers [1]. This theorem can be used to obtain the possible outputs of a comparison network and to deduce important properties that are necessary for understanding whether it is a sorting network or whether it is subsumed in another network.

The problem that we have tried to solve consists in finding an optimal size sorting network for a given number n of wires. Since the problem of finding an optimal sorting network of size k necessitates checking for the existence of all sorting networks with size smaller than k and the number of possible configurations grows by $\binom{n}{2}$ for each new comparator added, the problem is in co-NP [2]. Because of this, much of the research done in this domain focuses on finding lower and upper bounds for the size of sorting networks with n wires and in using heuristic algorithms in order to reduce the search space sorting networks.

2 Algorithm

In order to search for optimal size sorting networks, we have implemented the algorithm described in [3]. The best-first search process that is at the core of the strategy distinguishes between impractical paths and promising paths by using the evaluation function:

$$f(C) = \frac{1}{(n+1)(2^n-1)}(2^n \cdot |\mathrm{bad}(C)| + |\mathrm{outputs}(C)| - n - 1),$$

where outputs(C) are all the outputs of a given comparator network and bad(C) gives the postitions of the misplaced elements of the output of a sorting network.

The searching procedure can start either from an empty prefix or an initial comparator network on which additional comparators are added until a sorting network (a comparator network C for which $f(C) = 0$) is found. One notable prefix that can be added to our comparator networks is the Green filter, which can significantly decrease the number of computations at the expense of excluding some optimal solutions.

Another method which can be used to reduce the search space even further is to remove networks which are subsumed by others in our candidate set.

Let C and C' be two comparator networks. We say that C subsumes C' if there exists a permutation π of the set $\{1, 2, \cdots, n\}$ s.t. $\pi(\mathrm{outputs}(C)) \subset \mathrm{outputs}(C')$. If there is a subsumption relation between two networks, then the outputs of the two networks are identical [4]. This allows us to conclude that we can eliminate networks that are subsumed by others without losing any optimal solutions. Unfortunately, subsumption checking itself is a problem that is expensive in terms of computational time because we have to enumerate all $n!$ permutations for each pair that needs to be done. In order to counter this, first the permutation is modeled as a bipartite matching problem and then an algorithm is executed in order to find a perfect matching. This perfect matching represents the permutation that confirms the subsumption.

3 Experiments

The experiments were conducted on a regular computer (Intel i7-4720HQ @ 2.60Ghz), but no multithreading was used in order to speed-up the network generation and subsumption checking procedures. Almost all experiments, with the exception being $n = 6$ with the empty prefix, were run 15 times.

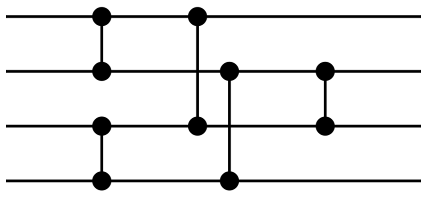
n	size	$\mathbf{t_{empty}}$ (s)	$\sigma_{\mathbf{empty}}$ (s)	$\mathbf{t_{Green}}$ (s)	$\sigma_{\mathbf{Green}}$ (s)
2	1	$5.5 \cdot 10^{-5}$	$2.25 \cdot 10^{-6}$	$1.21 \cdot 10^{-5}$	$1.84 \cdot 10^{-7}$
3	3	$1.87 \cdot 10^{-1}$	$8.65 \cdot 10^{-3}$	$1.56 \cdot 10^{-4}$	$6.34 \cdot 10^{-6}$
4	5	$9.33 \cdot 10^{-1}$	$2.81 \cdot 10^{-3}$	$4.29 \cdot 10^{-4}$	$6.16 \cdot 10^{-6}$
5	9	17.21	1.09	1.22	$3.40 \cdot 10^{-3}$
6	12	2311.87	-	5.97	$5.08 \cdot 10^{-2}$
7	16	-	-	40.43	1.98
8	19	-	-	81.12	4.30

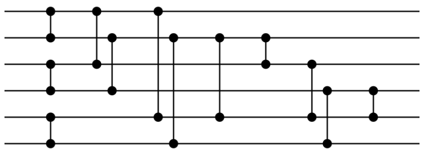
Table 1: Comparisons of execution times for finding the minimum size sorting network for $n = 2 \ldots 8$. t_{empty} is the mean execution time with no prefix and t_{green} is the mean executuion time with a Green filter prefix; they are presented along with their corresponding standard deviations. The second and third columns of the last two lines are not completed because the execution time was too large to be measured.

The resulting optimal sorting networks for $n = 4$, 6 respectively 8 are the following:

```
((0, 1), (2, 3), (0, 2), (1, 3), (1, 2))
((0, 1), (2, 3), (4, 5), (0, 2), (1, 3), (0, 4), (1, 5), (1, 4), (1, 2), (2, 4), (3, 5), (3, 4))
((0, 1), (2, 3), (4, 5), (6, 7), (0, 2), (4, 6), (1, 3), (5, 7), (0, 4), (1, 5), (2, 6), (3, 7), (2, 4), (1, 4), (5, 6), (3, 5), (3, 4), (1, 2), (5, 6))
```

and they are represented visually in Figure 1 on page 4.


(a) $n = 4$


(b) $n = 6$

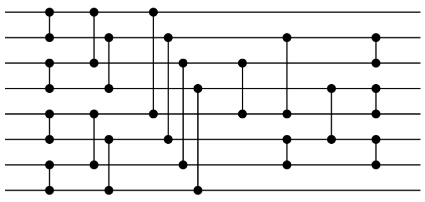

(c) $n = 8$

Figure 1: Sorting networks discovered by the best-first search algorithm.

References

[1] Cormen, T., Leiserson, C., Rivest, R., 1990. Introduction to algorithms. MIT Press.

[2] Parberry, I., 1991. On the computational complexity of optimal sorting network verification.

[3] Frasinaru, C., Raschip, M., 2019. Greedy Best-first Search for the Optimal-Size Sorting Network Problem

[4] Frasinaru, C., Raschip, M., 2017. An improved subsumption testing algorithm for the optimal-size sorting network problem.