Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

E. Kranakis, D. Krizanc

Computing boolean functions on Cayley networks
(Extended abstract)

Computer Science/Department of Algorithmics and Architecture Report CS-R9061 October

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Research (N.W.O.).

Computing Boolean Functions on Cayley Networks
(Extended Abstract)

Evangelos Kranakis [1] (eva@cwi.nl)

Danny Krizanc [2] (krizanc@cs.rochester.edu)

(1) Centrum voor Wiskunde en Informatica (CWI)
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

(2) University of Rochester, Department of Computer Science
Rochester, New York, 14627, USA

Abstract

We study the bit-complexity (i.e. total number of bits transmitted) of computing boolean functions on anonymous Cayley networks. We show that if G is a set of generators for a group $\mathcal{G}$ then a boolean function f is computable on the naturally labeled Cayley network $\mathcal{N}_G[\mathcal{L}_G]$ if and only if f is invariant under all automorphisms of the network. We also give efficient algorithms for computing boolean functions in all such networks. We give an algorithm that shows that for any group $\mathcal{G}$ and any set G of generators of $\mathcal{G}$ the bit complexity of computing all boolean functions which are computable on $\mathcal{N}_G[\mathcal{L}_G]$ is $O(|\mathcal{G}|\cdot\log^2|\mathcal{G}|\cdot\delta^2\cdot\sum_{g\in G}|g|^2)$, where δ is the diameter of the network, and $|g|$ the order of g in $\mathcal{G}$. In addition for any group $\mathcal{G}$ there is a set G of generators of $\mathcal{G}$ such that the above bit-complexity is $O(|\mathcal{G}|^3\cdot\log^4|\mathcal{G}|)$. Our results give even better complexity bounds for several well-known networks: rings, tori, hypercubes, star-, pancake- and bubble-sort networks. We also give a characterization of those abelian groups $\mathcal{G}$ which have a canonical set of generators G such that the network $\mathcal{N}_G$ computes more boolean functions than the network $\mathcal{N}_G[\mathcal{L}_G]$.

1980 Mathematics Subject Classification: 68Q99
CR Categories: C.2.1
Key Words and Phrases: Anonymous Networks, Cayley Networks, Boolean function, Group of automorphisms, Labeled and unlabeled networks.
Note: This paper will be submitted for publication elsewhere.

Report CS-R9061
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

1 Introduction

A very important problem in distributed computing is the designing of efficient protocols for computing functions in distributed networks of processors. For both practical and theoretical reasons it is useful to minimize the total number of exchanged bits which are necessary in order to compute certain functions. In this paper we consider the bit complexity of computing boolean functions on the class of Cayley networks (which includes the hypercube, the ring, the torus, the star networks, etc.) where the processors are identical and anonymous. We present a simple group-theorectic characterization of the boolean functions computable on a given Cayley network and also give an efficient algorithm for computing all such functions. For the case of many networks of interest, the algorithm is the most efficient known.

A distributed network is a simple, connected graph consisting of nodes (vertices) on which the processors are located, and links (edges) along which the interprocess communication takes place. The processors are assumed to have unlimited computational power but may exchange messages only with their neighbors in the network. Initially, each processor is given an input bit, 0 or 1.

The processors follow a deterministic protocol (or algorithm). During each step of the protocol they perform certain computations depending on their input value, their previous history and the messages they receive from their neighbors and then transmit the result of this computation to some or all of their neighbors. After a finite number of steps, predetermined by the initial conditions and the protocol, the processors terminate their computation and output a certain bit. Let B_N be the set of boolean functions on N variables. Let $\mathcal{N} = (V, E)$ be a network of size N, with node set $V = \{0, 1, \ldots, N-1\}$ and edge set $E \subseteq V \times V$. An input to $\mathcal{N}$ is an N-tuple $I =< b_v : v \in V >$ of bits $b_v \in \{0, 1\}$, where processor v receives as input value the bit b_v. Given a function $f \in B_N$ known to all the processors in the network we are interested in computing the value $f(I)$ on all inputs I. To compute f on input $I =< b_v : v \in V >$ each processor $v \in V$ starting with the input bit b_v should terminate its computation according to the given protocol and output the value b such that $f(I) = b$. A network computes the function f if for each input I, at the end of the computation each processor computes correctly the value $f(I)$. The bit complexity for computing f is the total number of bits exchanged during the computation of f. We are interested in providing algorithms that minimize the bit complexity of boolean functions.

We make the following assumptions regarding the networks and their processors (see also [Ang80]):

1. the processors know the network topology and the size of the network (i.e. total number of processors),

2. the processors are anonymous (i.e. they do not know either the identities of themselves or of the other processors),

3. the processors are identical (i.e. they all run the same algorithm),

4. the processors are deterministic,

5. the network is asynchronous,

6. the network links are labeled (i.e., there is a global, consistent labeling of the network links),

7. the network links are FIFO.

The class of networks we consider in this paper is that of the appropriately labeled Cayley networks. Intuitively, we represent the set of N processors by the elements of a group $\mathcal{G}$ of size N. We select a set G of generators of the group $\mathcal{G}$ and link two vertices if one can be obtained from the other by multiplication to the the right with a generator. Further, we label the links of the network by elements of G. Thus the processors are connected in a network of degree $|G|$ and size $|\mathcal{G}|$. Many standard network topologies (eg., rings, tori, hypercubes, star-graphs, etc.) can be obtained via a particular choice of $\mathcal{G}$ and G. The complexity of the algorithms we will develop will depend on the choice of the group and the set of generators we select.

The computability problem for an anonymous network $\mathcal{N}$ on N nodes is the following: Which boolean functions $f \in B_N$ are computable on the network $\mathcal{N}$? If as measure of complexity we consider the total number of bits transmitted in computing any boolean function f then this gives rise to the problem of finding an algorithm with optimal bit complexity. It is the purpose of this paper to study efficient algorithms for computing boolean functions on anonymous, labeled Cayley networks and in doing so we will give a characterization of the boolean functions which are computable in this network via its underlying group of automorphisms.

In its most general formulation, the computability problem was first considered for arbitrary unlabeled anonymous networks by [YK88] who gave a characterization in terms of "path trees". (However, the resulting algorithm was exponential in N^2.) Let f be a boolean function on N variables and let $S(f)$ be the group of permutations on N letters that leave f invariant on all inputs. Let $Aut(\mathcal{N})$ be the group of automorphisms of the network $\mathcal{N}$. For the following networks it is known that a boolean function f is computable on the N-node network $\mathcal{N}$ if and only if $S(f) \geq Aut(\mathcal{N})$: the complete network [YK88], the ring (oriented or not) [ASW85], the oriented d-dimensional torus [BB89], the oriented hypercube [KK90]. In section 3 we generalize these results by giving a characterization of the boolean functions computable on a labeled Cayley network in terms of the group of automorphisms of the network.

The simplest topology considered in the study of the bit complexity of computing boolean functions is the ring e.g., [AAHK88], [ASW85], [AS88], [MW86], [PKR84]. It has been shown by [ASW85] that there is an algorithm for computing all boolean functions which are computable on the ring, with bit complexity $O(N^2)$. In addition, [MW86] show that any nonconstant function has bit complexity $\Omega(N \cdot \log N)$ on the ring, and also construct boolean functions with bit complexity $\Theta(N \cdot \log N)$ on the ring. For the canonically labeled torus [BB89] give an algorithm with bit complexity $O(N^{1.5})$, and construct nonconstant functions with bit complexity $\Theta(N)$. Their results generalize to n-dimensional tori, resulting in an algorithm with bit complexity $O(N^{1+1/n})$, but the constant term implied by the O-notation depends on n. The case of the canonically labeled hypercube network on N nodes is studied in [KK90] where an algorithm with bit complexity $O(N \log^4 N)$ is given. In section 4 an efficient (polynomial bit complexity) algorithm is given for computing boolean functions on an arbitrary Cayley network. For a number of interesting cases, including the n-dimensional torus with non-constant n,

the hypercube, the n-star network, etc. the resulting algorithm is the most efficient known.

In the last section we study the impact of labeling a Cayley network on the class of functions which are computable in the network and show that for a well defined class of groups the labeling introduced is strong in the sense that the labeled network can compute boolean functions which cannot be computed in the unlabeled network.

1.1 Some Notation

In the sequel we use the following notation:

- We denote by $< G >$ the group generated by the set G of generators.
- e denotes the identity element of the group.
- For $u \in \mathcal{G}$, $|u|$ denotes the order of u in the group $\mathcal{G}$, i.e. the smallest positive integer k such that $u^k = e$.
- $\mathcal{G}_1 \otimes \mathcal{G}_2$ denotes the direct product of the groups $\mathcal{G}_1, \mathcal{G}_2$.

2 Cayley Networks

In this section we give the formal definition of Cayley network and provide a set examples of typical Cayley networks. Let $\mathcal{G}$ be a group and let $G \subseteq \mathcal{G}$ be a set of generators of $\mathcal{G}$. The Cayley network $\mathcal{N}_G$[1] of $\mathcal{G}$ with generators G is defined as follows: the set of vertices is $\mathcal{G}$, and the set of edges E is defined by

$$E = \{(u, v) : u^{-1}v \in G\}.$$

We assume that $G = G^{-1}$, where G^{-1} is the set of g^{-1} such that $g \in G$. To avoid loops in the network $\mathcal{N}_G$ we assume $e \notin G$. Further if $g = g^{-1}$ then we identify the edges g and g^{-1}.

We consider the following natural labeling $\mathcal{L}_G$ on the Cayley network $\mathcal{N}_G$: the label of the edge (u, v) is $u^{-1}v$. We denote the resulting network $\mathcal{N}_G[\mathcal{L}_G]$. By an automorphism ϕ of the labeled Cayley network we mean that the edge-labels are preserved under ϕ, i.e. if $(u, v) \in E$ then

$$\mathcal{L}_G(u, v) = \mathcal{L}_G(\phi(u), \phi(v)). \tag{1}$$

The group of automorphisms of $\mathcal{N}_G$ satisfying equation (1) is denoted by $Aut(\mathcal{N}_G[\mathcal{L}_G])$. Clearly every Cayley network is vertex transitive, in the sense that for any nodes $u, v \in \mathcal{G}$ there is a label preserving automorphism ϕ of $\mathcal{N}_G$ such that $\phi(u) = v$. The desired automorphism is

$$x \rightarrow \phi(x) = vu^{-1}x.$$

In fact this automorphism is uniquely determined from u and v, which makes the action of $Aut(\mathcal{N}_G[\mathcal{L}_G])$ on the vertices of $\mathcal{N}_G$ regular [Wie64]. The Cayley network has $|\mathcal{G}|$ nodes and the degree of each node is $|G|$, denoted by $d(G)$.

[1] A more correct notation would be $\mathcal{N}_{G,\mathcal{G}}$. But since the group $\mathcal{G}$ will be uniquely determined by the set of generators, $\mathcal{G} =< G >$, we avoid mentioning $\mathcal{G}$ in the symbol $\mathcal{N}_G$.

For each $g \in \mathcal{G}$ let $[g]_G$ = least number (with possible repetitions) of generators from G needed to represent g. Then the diameter of the labeled Cayley network $\mathcal{N}_G$ denoted by $\delta(G)$ is

$$\delta(G) = \max\{[g]_G : g \in \mathcal{G}\}.$$

The resulting Cayley network depends on the set G of generators we choose. One extreme is to choose $G = \mathcal{G}$ in which case $\mathcal{N}_G$ is the complete network on $|\mathcal{G}|$ nodes. At the other extreme, and this is the case we are usually interested in, we want to have "small" sets of generators with not "too big" diameter. In fact, as our subsequent studies indicate, we will be interested in sets G of generators minimizing the quantity $\delta(G)^2 \cdot \sum_{g \in G} |g|^2$. Small sets of generators exist in view of the well-known fact that every (finite) group $\mathcal{G}$ has a set of generators of size $O(\log |\mathcal{G}|)$. For a discussion of recent results on low diameter networks with small degree see [BHK+90]. Such groups have been found to be of great practical importance in the design of parallel and distributed networks. (See for instance [AK89], [AB90], as well as many others.) For another indication of the practicality of using such representations the reader is referred to [FHL80].

2.1 Some Examples

Throughout this section N denotes the number of nodes or the size of the corresponding group of the Cayley network. To simplify notation we list the elements of G without their inverses and we assume that multiplication of permutations is to the left.

First we consider examples arising from cyclic, abelian and dihedral groups.

Network	Group	Generators
Oriented Ring	C_N	$(1, 2, \ldots, N)$
Double Ring	D_n	$(1, 2, \ldots, n), \rho_n$
d-Torus	$(C_n)^d$	as usual
n-Hypercube	$F_n (\cong Z_2^n)$	$\phi_{\{1\}}, \phi_{\{2\}}, \ldots, \phi_{\{n\}}$

With cyclic groups we obtain a variety of tori. In oriented rings we have that the group of automorphisms is the cyclic group C_N which is generated by the N-cycle $(1, 2, \ldots, N)$. The d-dimensional torus is composed of rings of size n, where $N = n^d$. Its group of automorphisms is

$$(C_n)^d = \overbrace{C_n \otimes \cdots \otimes C_n}^{d}. \tag{2}$$

The dihedral group D_n is generated by $(1, 2, \ldots, n)$ and the reflection permutation

$$\rho_n = \begin{pmatrix} 1 & 2 & \cdots & n \\ n & n-1 & \cdots & 1 \end{pmatrix}.$$

The Cayley network arising from D_n has $N = 2n$ nodes, for $n \neq 2$. It consists of two rings each of size n and can be visualized best as follows. Place the rings horizontally in parallel on top of each other and connect the nodes of the top ring with the corresponding nodes of the bottom ring with n parallel vertical edges. The links within the rings are all labeled σ_n while the vertical links are labeled ρ_n. If we identify the paired nodes the resulting network is computationally equivalent to an unlabeled ring. (See [ASW85].) The group of automorphisms of the n-dimensional hypercube is the group F_n of bit-complement automorphisms. This is isomorphic to the group Z_2^n. The number of nodes

is $N = 2^n$. The generators of F_n we will consider are $\phi_{\{1\}}, \phi_{\{2\}}, \ldots, \phi_{\{n\}}$, where $\phi_{\{i\}}(x)$ is the sequence of bits obtained from x by complementing the ith bit of x.

Next we consider several examples arising by considering different generators of the symmetric group S_n, all having $N = n!$ nodes [AK89].

Network	Group	Generators
n-Star	S_n	$(1,k) : 1 < k \leq n$
n-Bubble-Sort	S_n	$(k-1,k) : 1 < k \leq n$
n-Pancake-Sort	S_n	$\rho_k : 1 < k \leq n$

The group of automorphisms of the n-star network is the symmetric group S_n on n letters. The generators of S_n we consider are the transpositions $(1,2),(1,3),\ldots,(1,n)$. It easy to see that

$$(i,j) = (1,i)(1,j)(1,i).$$

Every permutation on n letters is the product of $O(n)$ transpositions as above. Hence the diameter of the network is $\Theta(n)$. The group of automorphisms of the n-bubble network is the symmetric group S_n on n letters. The generators of S_n we will consider are the transpositions $(1,2),(2,3),\ldots,(n-1,n)$. It easy to see that

$$(i,j) = (i,i+1)(i+1,j)(i,i+1).$$

Every permutation on n letters is the product of $O(n^2)$ transpositions as above. The diameter of the network is $O(n^2)$. The automorphism group of the n-pancake network is generated by the reflections $\rho_2, \rho_3, \ldots, \rho_n$. Since $\rho_{i-1}\rho_i = (1,2,\ldots,i)$, $\rho_2 = (1,2)$ and

$$(i,i+1) = (1,2,\ldots,i)^{-i+1}(1,2)(1,2,\ldots,i)^{i-1}$$

these generators generate the symmetric group S_n. The diameter of the network is $O(n^2)$.

3 Automorphisms and Computability

The study of the bit complexity of computing boolean functions on Cayley networks is strongly interwoven with the group of automorphisms of the Cayley network.

3.1 Automorphisms of Cayley Networks

Before proceeding any further we need some results on the automorphism groups of Cayley networks. Throughout we assume that G is a set of generators for the group $\mathcal{G}$. We can prove the following theorem.

Theorem 3.1 *The group of automorphisms of $\mathcal{N}_G[\mathcal{L}_G]$ is isomorphic to $\mathcal{G}$.*

Proof. First we prove $\supseteq$. It is easy to show that for each $g \in \mathcal{G}$ $\phi_g \in Aut(\mathcal{N}_G[\mathcal{L}_G])$, where ϕ_g is the automorphism $\phi_g(u) = gu$. Indeed, assume $E(u,v)$ is true. Then

$$\phi_g(u)^{-1}\phi_g(v) = u^{-1}g^{-1}gv = u^{-1}v \in G.$$

Next we prove the other direction $\subseteq$. Let ϕ be an arbitrary automorphism of $\mathcal{N}_G$. We show that $\phi = \phi_g$, where $g = \phi(e)$ and e is the identity element of $\mathcal{G}$. Indeed, let

$u \in \mathcal{G}$. Then $u = g_1 g_2 \cdots g_k$, where $g_1, g_2, \ldots, g_k \in G$. Put $u^{(i)} = g_1 \cdots g_i$, if $i \geq 1$ and $u^{(i)} = e$ if $i = 0$. It is now easy to check that $u = u^{(k)}$ and each $(u^{(i-1)}, u^{(i)})$ is an edge of $\mathcal{N}_G$ with corresponding label g_i. Since ϕ is a label-preserving automorphism we have that g_i is also the label for the edge $(\phi(u^{(i-1)}), \phi(u^{(i)}))$. Hence $\phi(u^{(i)}) = \phi(u^{(i-1)})g_i$. It follows that $\phi(u) = gu$, as desired. Since $\phi_g \circ \phi_h = \phi_{gh}$, where $g, h \in G$ we have a natural isomorphism between $\mathcal{G}$ and $Aut(\mathcal{N}_G[\mathcal{L}_G])$. ∎

It is important to note at this point that Cayley networks can be endowed with natural labelings. They can also be characterized as those transitive networks whose automorphism group has a regular transitive subgroup (see also [Big74][lemma 16.3]).

Theorem 3.2 *If the automorphism group $Aut(\mathcal{N})$ of the transitive network $\mathcal{N}$ has a regular transitive subgroup $\mathcal{G}$ then there is a labeling $\mathcal{L}$ on $\mathcal{N}$ such that $\mathcal{G} = Aut(\mathcal{N}[\mathcal{L}])$. Conversely, the group of automorphisms of every Cayley network has a regular transitive subgroup.*

Proof (Outline). The "conversely" part has already been proved, since $Aut(\mathcal{N}_G[\mathcal{L}_G])$ is a regular transitive subgroup of $Aut(\mathcal{N}_G)$. To prove the other direction let x_0 be an arbitrary but fixed vertex of the network. Define the labeling $\mathcal{L}$ as follows

$$\mathcal{L}(\phi(x_0), \psi(x_0)) = \phi^{-1}\psi(x_0),$$

where $\phi, \psi \in \mathcal{G}$ and the the nodes $\phi(x_0)$ and $\psi(x_0)$ are adjacent in $\mathcal{N}$. This labeling is well defined since $\mathcal{G}$ is regular and transitive. Since for any $\alpha \in \mathcal{G}$,

$$\mathcal{L}(\alpha(\phi(x_0)), \alpha(\psi(x_0))) = \phi^{-1}\alpha^{-1}\alpha\psi(x_0) = \mathcal{L}(\phi(x_0), \psi(x_0))$$

we obtain that $\mathcal{G} \leq Aut(\mathcal{N}[\mathcal{L}])$. Conversely let $\alpha \in Aut(\mathcal{N}[\mathcal{L}])$. For the vertex $\alpha(x_0)$ there is a vertex $\phi(x_0)$ such that $\alpha(x_0) = \phi(x_0)$ where $\phi \in \mathcal{G}$. We claim that $\alpha = \phi$. To prove this we show that for all vertices x,

$$\alpha(x) = \phi(x) \Rightarrow \alpha = \phi \text{ on } \mathcal{N}(x)$$

where $\mathcal{N}(x)$ is the set of neighbors of x. Now the claim $\alpha = \phi$ follows by induction. ∎

3.2 Computability Problem

We now proceed with our study of the computability problem. The bit complexity of the algorithms we will develop depends on the type of generators for the group $\mathcal{G}$ we consider. Let $G = \{g_1, \ldots, g_k, g_1^{-1}, \ldots, g_k^{-1}\}$ be a set of generators for a group $\mathcal{G}$ given in some fixed enumeration. For every $g \in \mathcal{G}$ there exist exponents $a_1, \ldots, a_k$, $b_1, \ldots, b_k$, etc, such that

$$g = (g_1^{a_1} g_2^{a_2} \cdots g_k^{a_k})(g_1^{b_1} g_2^{b_2} \cdots g_k^{b_k}) \cdots .$$

Clearly every group element has such a representation (simply add the missing generators with exponent 0). The least number of expressions of the form $g_1^{a_1} g_2^{a_2} \cdots g_k^{a_k}$ needed to represent g is called the depth of g in G and is denoted by $\text{depth}_G(g)$. Further we define the depth of the group $\mathcal{G}$ with respect to G by

$$\text{depth}_G(\mathcal{G}) = \max\{\text{depth}_G(g) : g \in \mathcal{G}\}.$$

Notice that in general, $1 \leq \text{depth}_G(\mathcal{G}) \leq \delta(G)$.

Theorem 3.3 *Let G be a set of generators for the group $\mathcal{G}$. A boolean function f on $|\mathcal{G}|$ variables is computable on the anonymous network $\mathcal{N}_G[\mathcal{L}_G]$ if and only if $S(f) \geq Aut(\mathcal{N}_G[\mathcal{L}_G])$. Moreover the bit complexity of computing any such function is*

$$O\left(|\mathcal{G}| \cdot \left(\prod_{g \in G} |g|\right)^{\text{depth}_G(\mathcal{G})}\right).$$

Proof (Outline). First we show the necessity of $S(f) \geq Aut(\mathcal{N}_G[\mathcal{L}_G])$. For any input $I =< b_u : u \in \mathcal{G} >$ and any $\phi \in Aut(\mathcal{N}[\mathcal{L}_G])$ let $\phi(I) =< b_{\phi(u)} : u \in \mathcal{G} >$. Since $\mathcal{N}$ is transitive, u's labeled "path tree" on input I (see [YK88]) is identical to $\phi(u)$'s "path tree" when the input is $\phi(I)$. Since all processors execute the same algorithm, given the same data, it follows that u and $\phi(u)$ must output the same value, i.e. $f(I) = f(\phi(I))$.

Let $G = \{g_1, \ldots, g_k, g_1^{-1}, \ldots, g_k^{-1}\}$. To show the sufficiency of $S(f) \geq Aut(\mathcal{N}_G[\mathcal{L}_G])$ we execute the following "input collection" algorithm on any given input I.

1. Each processor sends its bit to its neighbor along direction g_1. Then the processors append the bit they receive to the sequence of bits they have so far. This part of the algorithm is executed for $|g_1|$ steps. The resulting sequence of bits is of length $|g_1|$.

2. Each processor sends its sequence to its neighbor along direction g_2. Then the processors append the sequence they receive to the sequence of bits they have so far. This part of the algorithm is executed for $|g_2|$ steps. The resulting sequence of bits is of length $|g_1| \cdot |g_2|$.

3. The processors execute this input collection algorithm along directions $g_1, \ldots, g_k$. At the end, the resulting sequence of bits is of length $\prod_{g \in G} |g|$.

To distribute the information to all processors the above algorithm must be "pipelined" to the resulting output for $\text{depth}_G(\mathcal{G})$ times. Let I_u be the sequence of bits collected by processor u at the end of the execution of the above algorithm. We call I_u the view of processor u on (the initial) input I. Then processor u outputs the value $f(I_u)$. It is easy to see that for any pair of processors u, v their views I_u, I_v are automorphic images of each other. Since f is invariant under $Aut(\mathcal{N}_G[\mathcal{L}_G])$ all processors will output the same value. ∎

The bit complexity implied by theorem 3.3 is far from optimal. For example, for the case of the n-star network with $N = n!$ nodes and the previously given set of generators we have that the bit complexity of computing boolean functions is $O(N^{1+\log N/\log\log N})$. In section 4 we will give another algorithm which is more efficient for computing boolean functions on many Cayley networks of interest.

Theorem 3.3 raises the problem of determining sets of generators minimizing the quantity $\text{depth}_G(\mathcal{G})$. A result due to Erdös and Rényi ([ER65]) shows that for any group $\mathcal{G}$ there is a set $G = \{g_1, \ldots, g_k\}$ of generators such that each $u \in \mathcal{G}$ can be written in the form $u = g_1^{a_1} \cdots g_k^{a_k}$, where $a_1, \ldots, a_k \in \{0, 1\}$ and $k = O(\log |\mathcal{G}|)$. This implies that any finite group has a set of generators of size $O(\log |\mathcal{G}|)$, $\text{depth}_G(\mathcal{G}) = 1$ and diameter $O(\log |\mathcal{G}|)$. Consequently we obtain the following result.

Corollary 3.1 *For any group $\mathcal{G}$ there exist a set of generators such that a boolean function f is computable on the network $\mathcal{N}_G[\mathcal{L}_G]$ if and only if $S(f) \geq Aut(\mathcal{N}_G[\mathcal{L}_G])$ and the bit complexity of computing all such functions is $O(|\mathcal{G}|^{1+\log|\mathcal{G}|})$.* ∎

For the case of abelian groups the bit complexity of computing boolean functions can be substantially improved by a careful choice of the set of generators. We call a set G of generators for an abelian group $\mathcal{G}$ *canonical* if it is obtained by one of the following rules:

- $\mathcal{G} =< g >$ is a cyclic group and $G = \{g\}$.
- $\mathcal{G} = \mathcal{G}_1 \otimes \mathcal{G}_2$, $G = G_1 \cup G_2$, and G_1, G_2 are canonical sets of generators for the groups $\mathcal{G}_1$, $\mathcal{G}_2$, respectively.

Clearly all groups obtained via the above rules are abelian and every abelian group is obtained via the above rules. For an abelian group the set of canonical generators is such that

$$\prod_{g \in G} |g| = |\mathcal{G}| \quad \text{and} \quad \text{depth}_G(\mathcal{G}) = 1.$$

This gives immediately the following consequence of theorem 3.3.

Corollary 3.2 *If G is a canonical set of generators of an abelian group $\mathcal{G}$ then a boolean function f is computable in the network $\mathcal{N}_G[\mathcal{L}_G]$ if and only if $S(f) \geq Aut(\mathcal{N}_G[\mathcal{L}_G])$ and the bit complexity of computing all such functions is $O(|\mathcal{G}|^2)$.* ∎

4 Efficient Algorithms

In this section we give an efficient algorithm for computing boolean functions on the labeled Cayley networks $\mathcal{N}_G[\mathcal{L}_G]$. First we outline the proof when the group $\mathcal{G}$ is abelian and G is a canonical set of generators; the corresponding network is a type of torus. If $G = \{g_1, \ldots, g_n, g_1^{-1}, \ldots, g_n^{-1}\}$ and $m_i = |g_i|$ then the torus is composed of the oriented rings $R_{m_1}, \ldots, R_{m_n}$ and has $N = m_1 \cdots m_n$ nodes. For simplicity we also assume $m := m_1 = \cdots = m_n$.

Let $T_{m,n}$ denote the n dimensional torus composed of n copies of the oriented ring R_m of m vertices; the total number of vertices of this network is $N = m^n$. A vertex p consists of sequences $p_1 p_2 \cdots p_n$, where $0 \leq p_1, p_2, \ldots, p_n < m$. Let $C_m =< (1, 2, \ldots, m) >$ be the cyclic group of order m generated by the cycle $\sigma = (1, 2, \ldots, m)$. The automorphism group of this torus is isomorphic to the cartesian product of n copies of C_m. We represent automorphisms by n integers $1 \leq i_1, i_2, \ldots, i_n \leq m$ in the following way:

$$\phi_{i_1, i_2, \ldots, i_n}(p_1 p_2 \ldots p_n) = \sigma^{i_1}(p_1)\sigma^{i_2}(p_2) \cdots \sigma^{i_n}(p_n).$$

Following this notation it is clear that the n-dimensional hypercube Q_n is $T_{2,n}$. Our main theorem is the following.

Theorem 4.1 *There is an algorithm for computing every boolean function $f \in B_N$ (which is invariant under all torus automorphisms) on the canonically labeled torus $T_{m,n}$, $N = m^n$, with bit complexity $O(N^{1+2/n} \cdot \log^4 N / \log^3 m)$.*

Proof (Outline). By corollary 3.2 there is an algorithm for computing all boolean functions which are computable in the torus with bit-complexity $O(N^2)$. We show how to transform this into an efficient algorithm by using: (1) a leader election mechanism, and (2) a coding mechanism of the views. Let $I =< b_u : u \in T_{m,n} >$ be a given input to

the network. For each node u let I_u be u's view when the input is I. (Throughout this proof we use the notation established in the course of the proof of theorem 3.3.) Define the group $\mathcal{G}(I)$ as follows:

$$\mathcal{G}(I) = \{\phi \in Aut(T_{m,n}) : \forall u \in T_{m,n}(I_u = I \Rightarrow \phi(I_u) = I)\}.$$

The group $\mathcal{G}(I)$ acts regularly on the nodes of the torus $T_{m,n}$. Hence for each $u \in T_{m,n}$ the orbits $\{\phi(u) : \phi \in \mathcal{G}(I)\}$ have exactly the same size, namely $|Aut(T_{m,n})|/|\mathcal{G}(I)| = m^n/|\mathcal{G}(I)|$. The abelian group $Aut(T_{m,n})$ has exactly n generators and by a well-known theorem on the number of generators of subgroups of finitely generated abelian groups, every subgroup of $Aut(T_{m,n})$, in particular $\mathcal{G}(I)$, has $\leq n$ generators [Kur60][§20]. It follows that the string I representing the view can be coded with a string of length $m^n/|\mathcal{G}(I)|$ (one bit, which is also the common bit, for each orbit) and $\leq n$ torus automorphisms generating the group $\mathcal{G}(I)$. Since every automorphism can be coded with $n \cdot \log m$ bits the resulting code has length

$$O\left(\frac{m^n}{|\mathcal{G}(I)|} + n^2 \cdot \log m\right).$$

The algorithm proceeds by induction as follows. At the ith step of the algorithm we are considering m^{n-i} tori each of size m^i. Each of these tori consists of m subtori, each of size m^{i-1}, forming a ring of size m. Suppose we have elected leaders at the $i-1$th step. The leaders of the kth subtorus in this ring correspond to processors with lexicographically maximal views and they all have the same view, say I_k^i. Any processor can determine from its own view and its knowledge of the network topology what the view of each processor is. Hence it can determine what the lexicographically maximal view is as well as the position of the leaders relative to itself. The corresponding group of the kth subtorus in this ring is $\mathcal{G}_k(I_k^i)$. Then the number of leaders of the subtorus is $\leq |\mathcal{G}_k(I_k^i)|$ and the views of these leaders can be coded with a string of length $m^{i-1}/|\mathcal{G}_k(I_k^i)|$ plus $\leq i-1$ torus automorphisms in $Aut(T_{m,i-1})$. Hence by the previous argument each such encoded message uses a total of

$$O\left(\frac{m^{i-1}}{|\mathcal{G}_k(I_k^i)|} + (i-1)^2 \cdot \log m\right)$$

bits. We now explain the ith step of the main algorithm, $i = 1, 2, \ldots, n$.. For each $k = 1, 2, \ldots, m$ do the following:

1. Leaders of the kth torus compute the group $\mathcal{G}_k(I_k^i)$ of their view I_k^i and encode their view as above.

2. Leaders of the kth torus transmit their view along the ring to their neighbors in the $(k+1) \bmod m$ torus. Processors receiving this message transmit it to their leaders within their torus which are at a distance $\leq (i-1) \cdot m$. These leaders transmit this message to the next torus and so on around the ring.

3. At the end of the round in each subtorus some (but not necessarily all) "old" leaders have obtained the views of all the leaders in the ring. They decode the messages and elect leaders for the ith step among the ones with lexicographically maximal views.

4. Leaders inform all the processors in the ith torus of their location.

This completes the description of the algorithm. The second step is the one that contributes the most to the bit complexity of the algorithm. The total bit complexity at the ith step is "number of tori (m^{n-i})" $\times$ "number of leaders" $\times$ "transmitting view to leaders which are at distance $(i-1)\cdot m$" $\times$ "transmitting around the ring (a factor m)" $\times$ "length of encoded view" for a total of

$$m^{n-i}\cdot|\mathcal{G}_k(I_k^i)|\cdot(i-1)\cdot m\cdot m\cdot O\left(\frac{m^{i-1}}{|\mathcal{G}_k(I_k^i)|}+(i-1)^2\cdot\log m\right)$$

bits. Using the fact that $|\mathcal{G}_k(I_k^i)|\leq m^{i-1}$ and summing the above for $k=1,\ldots,m$ and $i=1,\ldots,n$ we obtain the desired result. ∎

Using similar ideas we can extend the previous theorem to arbitrary labeled Cayley networks. The main result is as follows.

Theorem 4.2 *Let G be a set of generators for the group $\mathcal{G}$. A boolean function f on $|\mathcal{G}|$ variables is computable in the anonymous network $\mathcal{N}_G[\mathcal{L}_G]$ if and only if $S(f)\geq Aut(\mathcal{N}_G[\mathcal{L}_G])$ and the bit complexity of computing any such function is*

$$O\left(|\mathcal{G}|\cdot\log^2|\mathcal{G}|\cdot\delta(G)\cdot\mathrm{depth}_G(\mathcal{G})\cdot\sum_{g\in G}|g|^2\right).$$

Moreover there exists a set G of generators for $\mathcal{G}$ for which the above bit complexity is

$$O(|\mathcal{G}|^3\cdot\log^4|\mathcal{G}|).$$

Proof (Outline). We use the leader election and coding technique of theorem 4.1 in order to transform the algorithm of theorem 3.3. First enumerate the set of generators as $G=\{g_1,\ldots,g_n,g_1^{-1},\ldots,g_n^{-1}\}$ and let $m_i=|g_i|$. Each generator g_i forms an oriented ring of size m_i. The algorithm is inductive: at the ith step we distribute the messages along the i-th ring R_{m_i}. As in the proof of theorem 4.1 the number of tori involved in the ith step is $|\mathcal{G}|/|<g_1,\ldots,g_i>|$, the number of leaders is $|\mathcal{G}_i(I_k^i)|$ where $\mathcal{G}_i(I_k^i)$ is a subgroup of $<g_1,\ldots,g_i>$ and $k=1,\ldots,m_i$. Recipient processors transmit encoded view to leaders at distance $\delta(G)$, who then transmit it around the ring thus giving a factor m_i. The length of the encoded view is $O(|<g_1,\ldots,g_i>|/|\mathcal{G}_i(I_k^i)|+\log^2|\mathcal{G}|)$. Summing this for $k=1,\ldots,m_i$ and $i=1,\ldots,n$ and iterating for $\mathrm{depth}_G(\mathcal{G})$ times we obtain the desired result. The second part of the theorem follows from the results of Erdös and Rényi ([ER65]) mentioned above. ∎

Depending on the choice of generators the above algorithm may perform significantly better than that of theorem 3.3. In fact this raises the question of determining sets of generators G which minimize the quantity $\delta(G)\cdot\mathrm{depth}_G(\mathcal{G})\cdot\sum_{g\in G}|g|^2$.

For the case of the n-dimensional torus and hypercube we have the following bit complexities.

Network	Bit Complexity
$T_{m,n}$	$O(N^{1+2/n}\cdot\log^4 N/\log^3 m)$
Q_n	$O(N\cdot\log^4 N)$

This generalizes the $O(N^{1+1/n})$ bit complexity algorithm of [BB89] which is valid only when n is constant.

For the n-star, bubble sort and pancake networks we have that

$$\sum_{g \in G} |g|^2 = O(|G|) = O(\log N / \log\log N).$$

Hence the resulting bit complexity is $O(N \cdot \log^3 N \cdot \delta \cdot \text{depth}_G(S_n)/\log\log N)$. This gives the following table of complexities.

Network	Bit Complexity
n-Star	$O(N \cdot \log^5 N/\log\log^3 N)$
n-Bubble-Sort	$O(N \cdot \log^7 N/\log\log^5 N)$
n-Pancake-Sort	$O(N \cdot \log^7 N/\log\log^5 N)$

5 Labeling Cayley Networks

In the sequel we consider the impact of labelings on the computability problem on a Cayley network. In particular we are interested on whether or not the introduction of the labeling $\mathcal{L}_G$ alters the class of functions which are computable in the network. More specifically, we call the labeling $\mathcal{L}_G$ strong if there is a boolean function on $N = |\mathcal{G}|$ variables which is computable in the network $\mathcal{N}_G[\mathcal{L}_G]$ but not computable in $\mathcal{N}_G$. A boolean function f represents a group $\mathcal{G}$ if $S(f) = \mathcal{G}$. Such groups are called representable [CK89]. Then the following theorem is immediate.

Theorem 5.1 *Let G be a set of generators for the group $\mathcal{G}$. If the group $Aut(\mathcal{N}_G[\mathcal{L}_G])$ is representable and $Aut(\mathcal{N}_G[\mathcal{L}_G]) < Aut(\mathcal{N}_G)$ then $\mathcal{L}_G$ is strong.* ∎

For an extensive study of representable groups on which theorem 5.1 is applicable the reader should consult [CK89]. But still theorem 5.1 leaves ananswered the question of which groups satisfy the condition $Aut(\mathcal{N}_G[\mathcal{L}_G]) < Aut(\mathcal{N}_G)$. We now proceed to study this problem.

5.1 Arbitrary Groups

If we ignore labels then it is clear that $Aut(\mathcal{N}_G)$ consists of all permutations ϕ of $\mathcal{G}$ such that for all $u, v \in \mathcal{G}$,

$$u^{-1}v \in G \Leftrightarrow \phi(u)^{-1}\phi(v) \in G.$$

Two more groups of automorphisms that will be useful in our subsequent study are defined as follows.

$$Aut^*(\mathcal{N}_G) = \{\phi \in Aut(\mathcal{N}_G) : \forall u \in \mathcal{G} \forall g \in G(\phi(u)^{-1}\phi(ug) \in < g >)$$

and

$$Aut^{**}(\mathcal{N}_G) = \{\phi \in Aut(\mathcal{N}_G) : \forall u, a \in \mathcal{G}(\phi(u)^{-1}\phi(ua) \in < a >).$$

It is clear that

$$Aut(\mathcal{N}_G[\mathcal{L}_G]) \leq Aut^{**}(\mathcal{N}_G) \leq Aut^*(\mathcal{N}_G) \leq Aut(\mathcal{N}_G).$$

Now we can prove the following theorem which establishes a sufficient condition for the network $\mathcal{N}_G[\mathcal{L}_G]$ to have more computational power than the network $\mathcal{N}_G$.

Theorem 5.2 *If $Aut^{**}(\mathcal{N}_G) \neq Aut(\mathcal{N}_G)$ then $\mathcal{L}_G$ is strong.*

Proof. Let $\phi \in Aut(\mathcal{N}_G) \setminus Aut^{**}(\mathcal{N}_G)$. Since $\phi \notin Aut^{**}(\mathcal{N}_G)$ there exists $u, a \in \mathcal{G}$ such that

$$\phi(ua) \neq \phi(u)a^k, \text{ for all } 1 \leq k < |a| .$$

Define a boolean function on inputs $< b_x : x \in \mathcal{G} >$ as follows.

$$f(< b_x : x \in \mathcal{G} >) = \begin{cases} 0 & \text{if } \forall x \in \mathcal{G}(b_x = b_{xa}) \\ 1 & \text{otherwise.} \end{cases}$$

It is easy to see that f is kept invariant by all automorphisms of $\mathcal{N}_G[\mathcal{L}_G]$, but this is not true for the above automorphism ϕ. To see this consider an input $< b_x : x \in \mathcal{G} >$ such that $\forall x \in \mathcal{G}(b_x = b_{xa})$ and $b_{\phi(u)} \neq b_{\phi(ua)}$. It follows that

$$0 = f(< b_x : x \in \mathcal{G} >) \neq f(< b_{\phi(x)} : x \in \mathcal{G} >) = 1.$$

This completes the proof of the theorem. ∎

Theorem 5.2 raises the following question: for which groups $\mathcal{G}$ and sets G of generators is it true that $Aut^{**}(\mathcal{N}_G) \neq Aut(\mathcal{N}_G)$ or even $Aut^{*}(\mathcal{N}_G) \neq Aut(\mathcal{N}_G)$? In the sequel we show how to construct groups and generators satisfying the condition $Aut^{*}(\mathcal{N}_G) \neq Aut(\mathcal{N}_G)$.

Theorem 5.3 *Assume G_i is a set of generators for the group $\mathcal{G}_i$, with $i = 1, 2$, $G = G_1 \cup G_2$, $G_1 \cap G_2 = \emptyset$ and $\mathcal{G} = \mathcal{G}_1 \otimes \mathcal{G}_2$. Then*

$$\exists i(Aut^{*}(\mathcal{N}_{G_i}) < Aut(\mathcal{N}_{G_i})) \Rightarrow Aut^{*}(\mathcal{N}_G) < Aut(\mathcal{N}_G).$$

Proof. Assume on the contrary that $Aut^{*}(\mathcal{N}_{G_1}) < Aut(\mathcal{N}_{G_1})$ but $Aut^{*}(\mathcal{N}_G) = Aut(\mathcal{N}_G)$. Let $\phi_1 \in Aut(\mathcal{N}_{G_1})$. Define ϕ as follows: $\phi(u_1u_2) = \phi_1(u_1)u_2$, where $u_1 \in \mathcal{G}_1$ and $u_2 \in \mathcal{G}_2$. Then for all $u \in \mathcal{G}$, with $u = u_1u_2$, $u_1 \in \mathcal{G}_1, u_2 \in \mathcal{G}_2$, and all $g \in G$ we have that

$$\phi(u)^{-1}\phi(ug) = \begin{cases} g & \text{if } g \in \mathcal{G}_2 \\ \phi_1(u_1)^{-1}\phi(u_1g) & \text{if } g \in \mathcal{G}_1 . \end{cases} \tag{3}$$

It follows that $\phi \in Aut(\mathcal{N}_G)$. Since $Aut(\mathcal{N}_G) = Aut^{*}(\mathcal{N}_G)$ equation (3) implies that $\phi_1(u_1)^{-1}\phi_1(u_1g) \in < g >$, for all $u_1 \in \mathcal{G}_1$ and $g \in G_1$. It follows that $\phi_1 \in Aut^{*}(\mathcal{N}_{G_1})$. Consequently, $Aut^{*}(\mathcal{N}_{G_1}) = Aut(\mathcal{N}_{G_1})$, which is a contradiction. This completes the proof of the theorem. ∎

In proving $Aut^{*}(\mathcal{N}_G) \neq Aut(\mathcal{N}_G)$ sometimes it will be useful to look at the group of automorphisms of the group $\mathcal{G}$. If we define $Aut^{*}(\mathcal{G})$ as the automorphisms of $\mathcal{N}_G$ satisfying the following condition

$$(u, v) \in E \Rightarrow \phi(\mathcal{L}_G(u, v)) = \mathcal{L}_G(\phi(u), \phi(v)) \tag{4}$$

then we have the following precise characterization of $Aut^{*}(\mathcal{G})$.

Theorem 5.4 *The automorphisms of the Cayley network $\mathcal{N}_G$ satisfying condition (4) are exactly the automorphisms ϕ of the group $\mathcal{G}$ satisfying $\phi(G) = G$.*

Proof. Let ϕ be an automorphism of the group $\mathcal{G}$ satisfying $\phi(G) = G$. It follows from [Big74][section 16] that ϕ is an automorphism of the corresponding unlabeled Cayley network and condition (4) is easily verified. For the other direction assume ϕ is an automorphism of the network $\mathcal{N}_G$ satisfying condition (4). Let $u \in \mathcal{G}$, $g \in G$ and put $v = ug$. Clearly, $\mathcal{L}_G(u, v) = g$. Consequently, by (4) $\mathcal{L}_G(\phi(u), \phi(ug)) = \phi(\mathcal{L}_G(u, ug)) = \phi(g)$. This implies that $\phi(ug) = \phi(u)\phi(g)$. Similarly, we can prove $\phi(e) = e$. It is now easy to show that ϕ is a group automorphism. ∎

It follows that if $\phi \in Aut^*(\mathcal{G})$ and $\phi(g) \notin < g >$, for some $g \in G$, then $\phi \notin Aut^*(\mathcal{N}_G)$. Thus using theorem 5.4 we can prove that $Aut^*(\mathcal{N}_G) \neq Aut(\mathcal{N}_G)$ for the star, bubble-sort and pancake-sort networks previously considered.

5.2 Abelian Groups

For canonical sets of generators of abelian groups we can give a complete characterization of those Cayley networks for which $\mathcal{L}_G$ is a strong labeling. In fact we can prove the following theorem.

Theorem 5.5 *Let $\mathcal{G}$ be a nontrivial abelian group which is not any of the 4 cyclic groups C_2, C_3, C_4, C_5. If G is a canonical set of generators for $\mathcal{G}$ then the labeling $\mathcal{L}_G$ is strong.*

Proof. The proof is in several lemmas. First we take care of cyclic groups.

Lemma 5.1 *If G is a canonical set of generators for the cyclic group $\mathcal{G} = C_N$ then $\mathcal{L}_G$ is strong exactly when $N \neq 2, 3, 4, 5$.*

Proof of the lemma. The automorphism group of $\mathcal{N}_G$ is the dihedral group D_N. Results in [CK89] show that for any $N = 2, 3, 4, 5$ and any boolean function f on N variables if $S(f) \geq C_N$ then also $S(f) \geq D_N$. Hence in this case results in [ASW85] show that if f can be computed in $\mathcal{N}_G$ then f can also be computed in $\mathcal{N}_G[\mathcal{L}_G]$. ∎

If $\mathcal{G}$ is abelian then a precise characterization of the group $Aut^*(\mathcal{N}_G)$ is possible.

Lemma 5.2 *If $G = \{g_1, \ldots, g_k, g_1^{-1}, \ldots, g_k^{-1}\}$ is a canonical set of generators for the abelian group $\mathcal{G}$ then $Aut^*(\mathcal{N}_G) = \bigotimes_{1 \leq i \leq k} D_{|g_i|}$.*

Proof of the lemma. First observe that a canonical set of generators is irreducible, where a set G of generators of a group $\mathcal{G}$ is called irreducible if

$$g, g' \in G \text{ and } g' \neq g, g^{-1} \Rightarrow g' \notin < g > .$$

Let $s = |\{1 \leq i \leq k : g_i^2 \neq e\}|$ and assume that

$$g_1 \neq g_1^{-1}, \ldots, g_s \neq g_s^{-1}, g_{s+1} = g_{s+1}^{-1}, \ldots, g_k = g_k^{-1}.$$

Let $\phi \in Aut^*(\mathcal{N}_G)$. We show that ϕ is uniquely determined from

$$\phi(e), \phi(g_1), \ldots, \phi(g_k).$$

Since G is irreducible there exist $a_1, \ldots, a_s \in \{-1, 1\}$ such that $\phi(g_i) = g_i^{a_i}$, for $i = 1, \ldots, s$. Then we can prove that for all $u \in \mathcal{G}$ if

$$u = g_1^{r_1} \cdots g_s^{r_s} g_{s+1}^{r_{s+1}} \cdots g_k^{r_k}$$

then

$$\phi(u) = \phi(e)g_1^{a_1 r_1} \cdots g_s^{a_s r_s} g_{s+1}^{r_{s+1}} \cdots g_k^{r_k}$$

(the proof is by induction on the exponents). Now the isomorphism claimed in the statement of the theorem is

$$\phi \rightarrow (\phi(e), a_1, \ldots, a_s).$$

This completes the proof of the lemma. ∎

Lemma 5.3 *If G is a canonical set of generators for the abelian group $\mathcal{G}$ such that there exist $g, g' \in G$ with $|g| = |g'|$ and $g' \notin < g >$ then $Aut^*(\mathcal{N}_G) \neq Aut(\mathcal{N}_G)$.*

Proof of the lemma. Easy since we can prove that there is an automorphism ϕ of the group $\mathcal{G}$ permuting g, g' but leaving the other generators fixed. ∎

Now we give the proof of the main theorem. Assume that $\mathcal{G} = C_{n_1} \otimes \cdots \otimes C_{n_k}$, with $n_1 \geq \cdots \geq n_k$. If $n_i = n_j$ for some $i \neq j$ then by lemma 5.3 and theorem 5.2 the labeling $\mathcal{L}_G$ is strong. Hence without loss of generality we may assume that $n_1 > \cdots > n_k$. If $k = 1$ then the theorem follows from lemma 5.1. Hence without loss of generality we may assume $k \geq 2$. Assume now that for some i, $n_i \notin \{2, 3, 4, 5\}$. By [CK89] all dihedral groups are representable and the groups C_n are representable exactly when $n \neq 3, 4, 5$. Hence there is a boolean function f such that

$$S(f) = D_{n_1} \otimes \cdots \otimes D_{n_{i-1}} \otimes C_{n_i} \otimes D_{n_{i+1}} \otimes \cdots \otimes D_{n_k}.$$

Since $n_i \neq 2$, we have that $C_{n_i} < D_{n_i}$, and hence

$$S(f) < D_{n_1} \otimes \cdots \otimes D_{n_{i-1}} \otimes D_{n_i} \otimes D_{n_{i+1}} \otimes \cdots \otimes D_{n_k} = Aut^*(\mathcal{N}_G) \leq Aut(\mathcal{N}_G).$$

It follows that f is not computable in the network $\mathcal{N}_G$.

The theorem has been proved for all abelian groups except for the following eleven: $\bigoplus_{n \in S} C_n$, where $S \subseteq \{2, 3, 4, 5\}$ and $|S| \geq 2$, which we now consider. For these groups we use the automorhism groups $Aut^{**}(\mathcal{N}_G)$ and prove the following claim.

Claim. $Aut^{**}(\mathcal{N}_G) = Aut(\mathcal{N}_G[\mathcal{L}_G])$, for any of the eleven abelian groups considered.

Proof of the claim. Let $\phi \in Aut^{**}(\mathcal{N}_G)$. Since the groups considered are abelian it is easy to see that there exists an integer $k \geq 0$ such that for all $u, a \in \mathcal{G}$,

$$\phi(ua) = \phi(u)a^k.$$

It is now easy to check that for the eleven groups considered this implies that we can choose $k = 1$. This proves the claim.

Since a theorem of Subidussi [Sub64] implies that $Aut(\mathcal{N}_G)$ is not abelian (the same result also follows directly from the next theorem 5.6 without refering to [Sub64]) it follows that $Aut(\mathcal{N}_G) \neq Aut^{**}(\mathcal{N}_G)$. This completes the proof of the theorem. ∎

At this point it is interesting to note two interesting facts without proof. If $Aut_e(\mathcal{N}_G)$ is the set of automorphisms of $\mathcal{N}_G$ fixing the identity element e of the group $\mathcal{G}$ then every $\phi \in Aut(\mathcal{N}_G)$ is of the form $a \cdot \psi$, for some $a \in \mathcal{G}, \psi \in Aut_e(\mathcal{N}_G)$ (and the same result holds for any of the groups $Aut^*(\mathcal{N}_G), Aut^{**}(\mathcal{N}_G)$). It is also a consequence of the definition of $Aut^{**}(\mathcal{N}_G)$ and the proof of lemma 5.2 that for a canonical set of generators of an abelian group $\mathcal{G}$, $Aut_e^{**}(\mathcal{N}_G) = Aut^*(\mathcal{G})$. We leave the details to the reader.

The 11 abelian groups $\bigoplus_{n \in S} C_n$, where $S \subseteq \{2, 3, 4, 5\}$ and $|S| \geq 2$ have a rather interesting behavior. Although theorem 5.6 implies that the networks $\mathcal{N}_G$ and $\mathcal{N}_G[\mathcal{L}_G]$

cannot "distinguish" the boolean functions they can compute from their automorphism groups alone, theorem 5.5 shows that in fact the labeled network $\mathcal{N}_G[\mathcal{L}_G]$ can compute more boolean functions than the unlabeled network $\mathcal{N}_G$.

Theorem 5.6 *If G is a canonical set of generators for the abelian group $\oplus_{n\in S} C_n$, where $S \subseteq \{2,3,4,5\}$ then*

$$Aut(\mathcal{N}_G) = Aut^*(\mathcal{N}_G) = \bigoplus_{n\in S} D_n.$$

Moreover,

$$\{f \in B_N : S(f) \geq Aut(\mathcal{N}_G)\} = \{f \in B_N : S(f) \geq Aut(\mathcal{N}_G[\mathcal{L}_G])\}.$$

Proof. In order to prove the theorem we need the following lemma.

Lemma 5.4 *Assume that $\mathcal{G} = \mathcal{G}' \otimes C_n$ is an abelian group, G' a canonical set of generators for $\mathcal{G}'$, $G = G' \cup \{v\}$ and $|v| = 3$ or $|v| = 5$. Moreover assume that*

1. *if $|v| = 3$ then for all $g \in G'$ $|g| \neq 3$,*
2. *if $|v| = 5$ then for all $g \in G'$ $|g| \neq 3, 5$.*

Then $Aut(\mathcal{N}_{G'\cup\{v\}}) = Aut(\mathcal{N}_{G'}) \otimes D_{|v|}$.

Proof of the lemma. Let $\phi \in Aut(\mathcal{N}_{G'\cup\{v\}})$ and suppose on the contrary that for some $a \in \mathcal{G}, u \in G'$ $\phi(av) = \phi(a)u$ (the case $\phi(av) = \phi(a)u^{-1}$ is similar). We will derive a contradiction.

First consider the case $|v| = 3$. We have that

$$\phi(a) = \phi(av^3) = \phi(av^2)u_1 = \phi(av)u_2u_1 = \phi(a)u_2u_1u,$$

for some $u_1, u_2 \in G' \cup \{v\}$. But this implies that

$$u_1u_2u = e. \tag{5}$$

If $u_1, u_2 \notin \{u, u^{-1}\}$ then equation (5) implies that $u = e$, which is a contradiction. If $u_1 \in \{u, u^{-1}\}$ while $u_2 \notin \{u, u^{-1}\}$ then either $u_1 = u$ which implies that $u^2u_2 = e$, or else $u_1 = u^{-1}$ which implies that $u_2 = e$; in both cases we get a contradiction. Finally if $u_1, u_2 \in \{u, u^{-1}\}$ then either $u_1 = u_2 = u$, in which case (5) implies $u^3 = e$ (contradicting the fact that v is the unique element of order 3) or $u_1 = u_2 = u^{-1}$, in which case (5) implies $u^{-1} = e$, or $u_1 = u, u_2 = u^{-1}$, in which case (5) implies $u = e$. This proves the lemma in the case $|v| = 3$.

Next consider the case $|v| = 5$. As before there exist $u_1, u_2, u_3, u_4 \in G' \cup \{v\}$ such that

$$u_1u_2u_3u_4u = e. \tag{6}$$

We consider five cases depending on whether or not $0, 1, 2, 3, 4$ generators among the u_1, u_2, u_3, u_4 are in the set $\{u, u^{-1}\}$. As before we use the fact that there is no generator in G' of order 3 or 5 to derive a contradiction. This proves the lemma. ∎

In view of lemma 5.4 it is enough to consider only groups of the form $C_m \otimes C_2$, with $m > 2$, as well as of the form $C_m \otimes C_4$, with $m > 4$. We show that in these cases as well $Aut(\mathcal{N}_G) = Aut^*(\mathcal{N}_G)$. This would imply that for all 16 abelian groups

$\otimes_{m \in S} C_m$, where $S \subseteq \{2,3,4,5\}$ and for any canonical set of generators G of that group $Aut(\mathcal{N}_G) = Aut^*(\mathcal{N}_G)$.

First consider the case of the groups $C_m \otimes C_2$, with $m > 2$. Let u be a generator of C_m and v a generator of C_2. Let $\phi \in Aut(\mathcal{N}_{\{u,v\}})$ and suppose on the contrary that $\phi(uv) = \phi(u)u$ (the case $\phi(uv) = \phi(u)u^{-1}$ is similar). We will derive a contradiction. It follows that $\phi(u^2v) = \phi(u)u_1u$, for some $u_1 \in \{u, u^{-1}, v\}$. If $u_1 = u$ then $\phi(u^kv) = \phi(u)u^k$, for all k. If $u_1 = u^{-1}$ then $\phi(u^kv) = \phi(u)u^{-k}$, for all k. If $u_1 = v$ then $\phi(u^kv) = \phi(u)u^{k-1}v$, for all k. But all these statements contradict the injectivity of ϕ.

It remains to examine the case of the abelian groups $\mathcal{G} = C_m \otimes C_4$ when $m > 4$. Let u, v be generators of C_m, C_4, respectively. By contradiction, assume that for some $a \in \mathcal{G}$, and $\phi \in Aut(\mathcal{N}_G)$, $\phi(av) = \phi(a)u$. But arguing as before this would imply that ϕ is not $1-1$. This completes the proof of theorem 5.6. ∎

Theorems 5.3, 5.4, 5.5 give a class of groups for which $Aut^*(\mathcal{N}_G) < Aut(\mathcal{N}_G)$. The interested reader can find more information on the automorphism group $Aut(\mathcal{N}_G)$ in [Cha64] and [Sub64] (see also the correction in [Imr69] and [Imr70]), where it is shown that $Aut(\mathcal{N}_G)$ is never abelian unless $\mathcal{G} = Z_2^n$ and $n \neq 2, 3, 4$.

References

[AAHK88] K. Abrahamson, A. Adler, L. Higham, and D. Kirkpatrick. Randomized evaluation on a ring. In Jan van Leeuwen, editor, *Distributed Algorithms, 2nd International Workshop, Amsterdam, The Netherlands, July 1987*, volume 312, pages 324 – 331. Springer Verlag Lecture Notes in Computer Science, 1988.

[AB90] F. Annexstein and M. Baumslag. A unified approach to off-line permutation routing on parallel networks. In *Proceedings of 2nd Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 398–406, 1990.

[AK89] S. B. Akers and B. Krishnamurthy. A group theoretic model for symmetric interconnection networks. *IEEE Transactions on Computers*, 38(4):555 – 566, 1989.

[Ang80] D. Angluin. Local and global properties in networks of processors. In *12th Annual ACM Symposium on Theory of Computing*, pages 82 – 93, 1980.

[AS88] H. Attiya and M. Snir. Better computing on the anonymous ring. Technical Report RC 13657 (number 61107), IBM T. J. Watson Research Center, November 1988. 33 pages.

[ASW85] C. Attiya, M. Snir, and M. Warmuth. Computing on an anonymous ring. In *4th Annual ACM Symposium on Principles of Distributed Computation*, pages 196 – 203, 1985.

[BB89] P. W. Beame and H. L. Bodlaender. Distributed computing on transitive networks: The torus. In B. Monien and R. Cori, editors, *6th Annual Symposium on Theoretical Aspects of Computer Science, STACS*, pages 294–303. Springer Verlag Lecture Notes in Computer Science, 1989.

[BHK+90] L. Babai, G. Hetyei, W. M. Kantor, A. Lubotzky, and Á. Seress. On the diameter of finite groups. In *Proceedings of IEEE 31th Annual Symposium on Foundations of Computer Science*, pages 857–865, 1990.

[Big74] N. Biggs. *Algebraic Graph Theory.* Cambridge University Press, 1974.

[Cha64] C-Y. Chao. On a theorem of Subidussi. *Proceedings American Mathematical Society*, 15:291 – 292, 1964.

[CK89] P. Clote and E. Kranakis. Boolean functions invariance groups and parallel complexity. In *4th Annual IEEE Symposium on Structure in Complexity Theory*, 1989. Full version to appear in SIAM journal on Computing.

[ER65] P. Erdös and A. Rényi. Probabilistic methods in group theory. *Journal d'Analyse Mathématique*, 14:127–138, 1965.

[FHL80] M. Furst, J. Hopcroft, and E. Luks. Polynomial-time algorithms for permutation groups. In *Proceedings of 21st Annual IEEE Symposium on Foundations of Computer Science, Syracusea, NY*, pages 36 – 41, 1980.

[Imr69] W. Imrich. Graphen mit transitiver Automorphismgruppe. *Monatshefte für Mathematik*, 73:341 – 347, 1969.

[Imr70] W. Imrich. Graphs with transitive abelian automorphisms groups. In P. Erdös, A. Renyi, and V. Sós, editors, *Combinatorial Theory and its Applications*, volume II. North Holland, 1970.

[KK90] E. Kranakis and D. Krizanc. Computing boolean functions on anonymous hypercube networks. Technical Report CS R90-40, Centrum voor Wiskunde en Informatica, Department of Algorithmics and Architecture, 1990. 12 pages.

[Kur60] A. G. Kurosh. *The Theory of Groups*, volume 1. Chelsea, 1960. 2nd english edition (translated from the Russian).

[MW86] S. Moran and M. Warmuth. Gap theorems for distributed computation. In *5th Annual ACM Symposium on Principles of Distributed Computation*, pages 131 – 140, 1986.

[PKR84] J. Pachl, E. Korach, and D. Rotem. A new technique for proving lower bounds for distributed maximum finding algorithms. *Journal of the ACM*, 31(4):905 – 918, October 1984.

[Sub64] G. Subidussi. Vertex transitive graphs. *Monatshefte für Mathematik*, 68:426 – 438, 1964.

[Wie64] H. Wielandt. *Finite Permutation Groups.* Academic Press, 1964.

[YK88] M. Yamashita and T. Kameda. Computing on an anonymous network. In *7th Annual ACM Symposium on Principles of Distributed Computation*, pages 117 – 130, 1988.