Optimal Gray-code Labeling and Recognition Algorithms for Hypercubes

Stavros D. Nikolopoulos

Department of Computer Science, University of Ioannina,
P.O. Box 1186, GR-45110 Ioannina, Greece
e-mail: stavros@cs.uoi.gr

Abstract — We present an optimal greedy algorithm which returns a Gray-code labeling of the nodes of an n-dimensional hypercube, that is, a labeling of the nodes with binary strings of length n for which the Hamming distance between two nodes is 1 if and only if these are adjacent in the hypercube. The proposed algorithm is very simple; it uses breadth-first search to guide the greedy chose of labels and computes the Gray-code label of a node u by performing the logical disjunction of the Gray-code labels of two nodes adjacent to node u. It takes as input a hypercube Q_n with $N = 2^n$ nodes and runs in $O(N\log N)$ time. Moreover, based on the labeling algorithm we propose a recognition algorithm which runs in $O(N\log N)$ time. Thus, in view of the fact that Q_n has $n2^{n-1}$ edges, this behaviour is optimal. Our algorithms have the property that they can be efficiently implemented in a PRAM model of computation.

Keywords: Hypercube, Gray-code labeling, recognition, graph partition, parallel algorithms, complexity.

1. Introduction

Hypercubes are extremely pervasive in the literature of contemporary computer science as they provide a structural model for parallel computer architectures. With the advances in VLSI technology, it has become feasible to build computing machines with hundreds or even thousands of processors cooperating in solving a given problem. These machines differ along various dimensions such as control mechanism, address-space organization, interconnection network, and granularity of processors. Shared memory and non-shared memory (message-passing) parallel machines can be constructed by connecting processors and memory units using a variety of interconnection networks.

Several topologies have been proposed for interconnecting the processors of a non-shared memory parallel computing system. Among them, due to its topological richness, the hypercube topology (also known as n-cube, Cosmic cube, Boolean cube) is extremely pervasive in the literature as it provides a structural model for parallel computer architectures (it offers a large bandwidth, logarithmic diameter, and a high degree of fault tolerance) [1, 16]. Both research and commercial systems have been build using the hypercube interconnection scheme, and significant research effort have been made on hypercube architectures [11, 13, 18].

We assume that a hypercube interconnection scheme is represented by its underlying graph, and let N and n be two integers such that $N = 2^n$. An n-dimensional hypercube $Q_n = (V_n, E_n)$ is an N-node graph which is defined recursively as the iterated Cartesian product of the smallest nontrivial complete graph K_2 consisting of two nodes and one edge joining them:

i) $Q_1 = K_2$, and

ii) $Q_n = K_2 \times Q_{n-1}$, for $n \geq 2$,

where $\times$ is the Cartesian product of two graphs [7]. Examples of Q_1, Q_2, Q_3 and Q_4 are shown in Fig. 1. It is convenient to say that $Q_0 = K_1$.

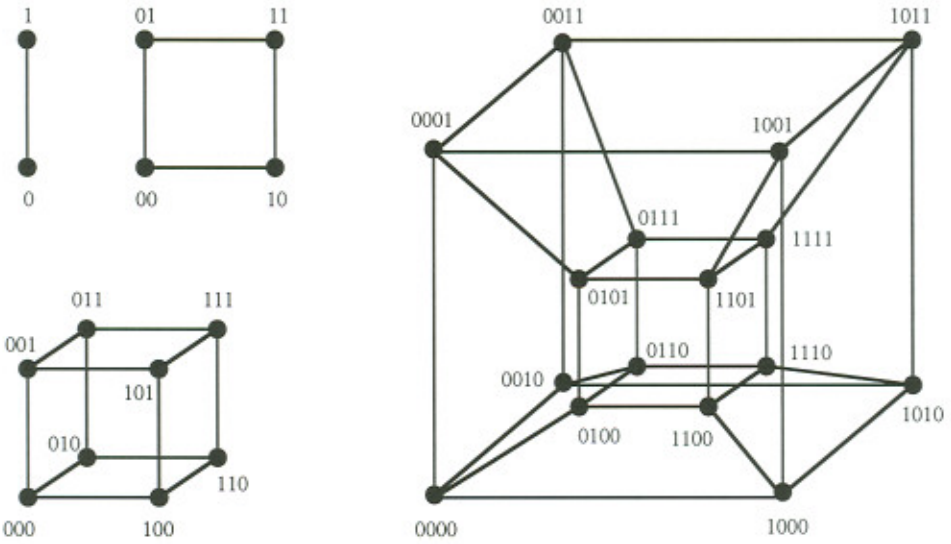

Fig. 1. The first four hypercubes Q_1, Q_2, Q_3 and Q_4.

Characterizations and topological properties of hypercubes have been extensively studied and interesting results have been reported in the literature [5, 10, 16, 17]. In the following, we describe some of the most important characterizations and topological properties of an n-dimensional hypercube Q_n relating to sparsity, diameter, existence of node disjoint parallel paths, and existence of odd and even cycles.

It is well-known that a characterization of a family $\mathcal{F}$ of graphs gives a necessary and sufficient condition for a given graph G to be in $\mathcal{F}$. For example, G is bipartite if and only if every cycle of G has even length [7]. We list three characterizations for a given graph G to be a hypercube Q_n. In principle, each of these conditions contains enough information to enable the logical deduction that G is indeed an n-cube graph or n-dimensional hypercube (we shall only list these criteria and include a reference for each, where the details may be found). The first condition was given by Foldes [5] (see Criterion C1). A similar but different condition (see Criterion C2) was derived by Garey and Graham [6] in their study of "squashed cubes". Combining their results with the Merger's Theorem [7, p. 47, Theorem 5.9], we easily conclude that the number of node-disjoint u-v paths in a Q_n is d. The third condition is due to Laborde and Hebbare [12].

(C1) A connected graph G is a hypercube if and only if G is bipartite and the number of geodesics between any two nodes at distance d is $d!$.

(C2) Graph G is some Q_n if and only if G is connected and bipartite and for any two nodes u, v at distance d, the connectivity $\varkappa(u, v) = d$.

(C3) A connected graph G with n nodes and minimum degree δ is a hypercube if and only if every pair of adjacent edges lie in an unique 4-cycle and $n = 2^{\delta}$.

Let us now focus on the topological properties of a hypercube Q_n. Some of them can be easily proved or can be immediately derived from characterizations C1 - C3. The property P4 is the most important.

(P1) Q_n is a connected bipartite graph.

(P2) Q_n has $n2^{n-1}$ edges.

(P3) The diameter of Q_n is $n = \log N$.

(P4) Each node in a Q_n can be uniquely represented by an n-bit label in such a way that two nodes are adjacent if and only if their labels differ in exactly one bit.

For convenience, we will number the bits in a label of a node of Q_n from right to left as 0 to n-1. That is, a binary string b of length n will be written as $b_{n-1}b_{n-2} \ldots b_1b_0$, where b_{n-1} is the most significant bit and b_0 is the least significant bit. The i-th bit (or bit i) of b is b_i for $0 \le i \le n$-1. Fig. 1 shows that the hypercubes Q_1, Q_2, Q_3 and Q_4 with a binary labeling of their nodes. If two adjacent nodes differ in their i-th bit, then they are said to be in *direction i* with respect to each other. For example, the node u with label 1011 is said to be in direction 2 of a Q_4 with respect to node v with label 1111 and vice versa. It is clear from this definition that there are n distinct directions in a hypercube Q_n.

We now consider the labels s and t of two nodes in a Q_n. The total number of bits positions at which these two labels differs called the *Hamming distance* between them [7, 11]. For example, the Hamming distance between nodes labeled 011 and 101 in a 3-dimensional hypercube Q_3 is 2. Based on the Hamming distance, it is easy to see that the n-dimensional hypercubes have the property that two nodes are adjacent if and only if their Hamming distance is 1. Furthermore, take the property P4 into account we can define the hypercube $Q_n = (V_n, E_n)$ as a graph where V_n is the set of all 2^n binary n-strings and E_n is precisely those pairs of n-strings whose labels vary in exactly one binary digit (see Fig. 2). This presentation is immediately seen to be equivalent (as it must!) to presentation in terms of Cartesian products.

Many problems have arisen in the study of binary strings that may be stated in terms of hypercube graphs. Perhaps the best known example is that of generating "Gray codes". We define a *Gray-code* as a sequence of 2^n binary n-string where successive strings are distinct in exactly one binary digit. Thus, a Gray-code is simply a Hamiltonian cycle on a hypercube and, hence, the number of Gray codes is the number of Hamiltonian cycles on Q_n.

A Gray-code labeling of a hypercube is a node labeling for which the Hamming distance between two nodes is 1 if and only if these are adjacent in the hypercube. Gray-code labeling is a fundamental issue of both theoretical and practical importance. Its most important application is related to the portability of algorithms across various parallel architectures. Specifically, with the widespread availability of parallel architectures based on the hypercube interconnection scheme,

there is an interest in the portability of algorithms developed for architectures based on other topologies, such as linear arrays, rings, two-dimensional meshes, and complete binary trees, into the hypercube. This question of portability reduces to one of embedding the above interconnection schemes into the hypercube. Gray codes (binary reflected Gray codes) have been extensively used in embedding arrays, rings, meshes and trees into the hypercube.

Our objective is to study the Gray-code labeling and recognition of an n-dimensional hypercube and propose optimal sequential and/or parallel algorithms for that problems. Many researchers have extensively studied hypercubes and proposed algorithms for many important problems among which the Gray-code labeling and recognition problems. Recently, Bhagavathi *et. al.* [2] proposed a greedy hypercube-labeling algorithm for Hypercubes on $N = 2^n$ nodes, which runs in $O(N \log N)$ time being, therefore, optimal. The main feature of their algorithm is that it uses depth-first search to guide the greedy choice of labels.

In this paper we first present a simple and optimal algorithm for Gray-code labeling of the nodes of an n-dimensional hypercube. Specifically, given a hypercube Q_n on $N = 2^n$ nodes, our labeling algorithm visits the nodes of the hypercube by using the breadth-first search and computes the Gray-code label of a node u by performing the logical disjunction of the Gray-code labels of two nodes adjacent to node u. It runs in $O(N \log N)$ time and, therefore, in view of the fact that Q_n has $n2^{n-1}$ edges, this behaviour is optimal. Based on the labeling algorithm we describe an optimal algorithm which recognizes whether a given graph on $N = 2^n$ nodes is indeed an n-dimensional hypercube. Our recognition algorithm runs in $O(N \log N)$ time being, therefore, optimal. Moreover, since our algorithms are based in the breadth-first search, they can efficiently implemented in a PRAM model of computation. More precisely, we present optimal parallel algorithms for the Gray-code labeling and recognition problems. Our main technique is based on the notion of partitioning the vertex set of a hypercube, with respect to a vertex s, into a set of (mutually disjoint) adjacency-level sets. We propose two parallel algorithms for Gray-code labeling of a hypercube Q_n; the first algorithm runs in $O(\log N)$ time using a total of $O(N \log N)$ operations on a CRCW PRAM model, while the second one is executed on a CREW PRAM in $O(1)$ time using $O(N \log N)$ operations, when the adjacency-levels sets of a partition of the vertex set of a hypercube are given. We also show that the recognition problem can be solved in $O(\log N)$ time using $O(N \log N)$ operations on a CRCW PRAM model.

The paper is structured as follows. Section 2 presents the main technical results and the key ideas that are at the hear of our optimal hypercube-labeling and recognition algorithms. The labeling and recognition algorithms and their complexity analyses are presented in Section 3 and Section 4, respectively. Parallel hypercube-labeling and recognition algorithms are described in Section 5. Finally, Section 5 summarizes our results.

2. The Main Results

Given a connected graph $G=(V,E)$ and a vertex $v \in V$, we define a partition $\mathcal{L}(G,v)$ of the vertex set V (we shall frequently use the term *partition of the graph* G), with respect to the vertex v as follows:

$$\mathcal{L}(G,v) = \{ N_i(v) \mid v \in V, 0 \le i \le L_v, 0 \le L_v < |V| \}$$

where $N_i(v)$, $0 \le i \le L_v$, are the *adjacency-level sets*, or simply the *adjacency-levels*, and L_v is the *length* of the partition $\mathcal{L}(G,v)$ [14]. The adjacency-level sets of the partition $\mathcal{L}(G,v)$ of the graph G,

are formally defined as follows:

$$N_i(v) = \{\, u \in V \mid d(v, u) = i,\ 0 \le i < |V| \,\}$$

where $d(v, u)$ denotes the *distance* between vertices v and u in G. We point out that $d(v, u) \ge 0$, and $d(v, u) = 0$ when $v = u$, for every $v, u \in V$. (In the case where G is a disconnected graph, $d(v, u) = \infty$ when v and u do not belong to the same connected component.) Obviously, $L_v = max\ \{d(v, u) \mid u \in V\}$, $N_0(v) = \{v\}$ and $N_1(v) = N(v)$. In Fig. 2 we illustrate the partitions of the graphs Q_3 and Q_4, with respect to the vertices v and u, respectively, where $v = 000$ and $u = 0000$.

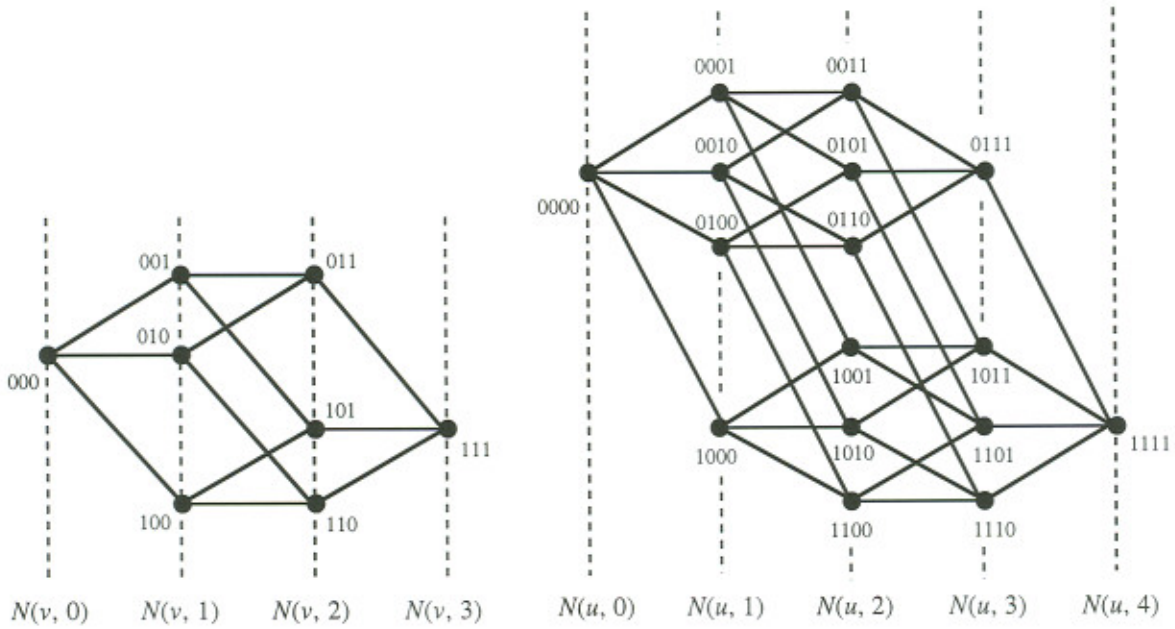

Fig. 2. Partition of the hypercubes Q_3 and Q_4, where $v = 000$ and $u = 0000$.

Let $G = (V, E)$ be a graph with n nodes and m edges. It is easy to see that, the adjacency-level sets $N_i(v)$, $0 \le i \le L_v$, of a partition $\mathcal{L}(G, v)$ can be computed in $O(n + m)$ time using breadth-first search [4]. Moreover, the adjacency-level sets of the partition $\mathcal{L}(G, v)$ can easily be computed recursively as follows: $N_i(v) = \{y \mid (x, y) \in E \text{ and } x \in N_{i-1}(v)\} - \{N_{i-1}(v) \cup N_{i-2}(v)\}$, $2 \le i \le L_v$. In a parallel process environment, these sets can be computed in $O(n)$ time with $O(n^2)$ processors using a CRCW PRAM. (We note that, these sets can also be computed by considering first the distance matrix of the graph G and then extracting all set information that is necessary. This computation can be efficiently done by using matrix multiplication; see [3].)

In the rest of the paper we shall denote the adjacency-level sets $N_0(v)$, $N_1(v)$, ..., $N_i(v)$ of a partition $\mathcal{L}(G, v)$ as $N(v, 0)$, $N(v, 1)$, ..., $N(v, i)$, $0 \le i \le L_v$.

Lemma 2.1 Let $Q_n = (V_n, E_n)$ be an n-dimensional hypercube on $N = 2^n$ nodes, and let $N(v, 0)$, $N(v, 1)$, ..., $N(v, n)$ be the adjacency-level sets of the partition $\mathcal{L}(Q_n, v)$, where $v \in V_n$. Then, $\langle N(v, i) \rangle$ is a mK_1 graph, where $m = |N(v, i)|$ and $0 \le i \le n$.

Proof. Immediately from criterion C1 (or criterion C2). □

Lemma 2.2 Let $Q_n = (V_n, E_n)$ be an n-dimensional hypercube on $N = 2^n$ nodes, and let $N(v, 0)$, $N(v, 1)$, ..., $N(v, n)$ be the adjacency-level sets of the partition $\mathcal{L}(Q_n, v)$, where $v \in V_n$. Let u, w be distinct vertices in $N(v, i)$, $1 \le i \le n$-1. The following statements mast be satisfied:

(a) Nodes u, w have at most one common neighbour in $N(v, i+1)$;

(b) Nodes u, w have a common neighbour in $N(v, i+1)$ if and only if they have a common neighbour in $N(v, i$-$1)$;

Proof. (a) Immediately from criterion C3. (b) Immediately from criterion C3. □

Construction properties of a hypercube: Let K_2 and Q_n be two hypercubes with 2 and $N = 2^n$ nodes, respectively, $n \ge 1$. Let $\{v_1, v_2\}$ be the node set of K_2 and let $\{u_1, u_2, ..., u_N\}$ be the node set of Q_n. By definition, $Q_{n+1} = K_2 \times Q_n$ is a hypercube with $2N = 2^{n+1}$ nodes. The node set $V(Q_{n+1})$ and the edge set $E(Q_{n+1})$ of the hypercube Q_{n+1} are the following:

(i) $V(Q_{n+1}) = \{x_1, x_2, ..., x_N, y_1, y_2, ..., y_N\}$, where $x_i = (v_1, u_i)$ and $y_i = (v_2, u_i)$, $1 \le i \le N$;

(ii) $(x_k, y_k) \in E(Q_{n+1})$, for $1 \le i \le N$, and
$(x_i, x_j), (y_i, y_j) \in E(Q_{n+1})$ if and only of $(u_i, u_j) \in E(Q_n)$, for $1 \le i, j \le N$ and $i \ne j$.

Let H_n and F_n be two subgraphs of the graph Q_{n+1} induced by $\{x_1, x_2, ..., x_N\}$ and $\{y_1, y_2, ..., y_N\}$, respectively, where $x_i = (v_1, u_i)$ and $y_i = (v_2, u_i)$, $1 \le i \le N$. Let $\mathcal{L}(H_n, x_1)$ and $\mathcal{L}(F_n, y_1)$ be two partitions of H_n and F_n, respectively, and let

$$N(x_1, 0), N(x_1, 1), ..., N(x_1, n)$$

and

$$N(y_1, 0), N(y_1, 1), ..., N(y_1, n)$$

be the adjacency-level sets of these partitions. Moreover, let $\mathcal{L}(Q_{n+1}, s)$ be a partition of Q_{n+1} with respect to node $s = x_1$, and let $N(s, 0)$, $N(s, 1)$, ..., $N(s, n+1)$ be the adjacency-level sets of $\mathcal{L}(Q_{n+1}, s)$. Then, the following hold:

(i) $N(s, 0) = N(x_1, 0)$,
$N(s, n+1) = N(y_1, n)$, and
$N(s, i) = N(x_1, i) \cup N(y_1, i$-$1)$, $1 \le i \le n$;

(ii) There is one and only one edge $(x_k, y_k) \in E(Q_{n+1})$ such that $x_k \in V(H_n)$ and $y_k \in V(F_n)$, $1 \le k \le N$ (see criterion C3). Moreover, if node $x_k \in N(s, i)$ then node $y_k \in N(s, i+1)$, where $0 \le i \le n$;

Next, we describe a Gray-code labeling method which, as we shall see later, will be used as a basis for the proposed Gray-code labeling algorithm. For simplicity, hereafter, we shall say that a hypercube is *Gray-code labeled* or *G-labeled* if the n-bit labels of its nodes satisfy the property P4. Moreover, we shall refer to the Gray-code label of a node u as *G-label(u)*.

Gray-code labeling of a hypercube: Let K_2 and Q_n be two hypercubes and let $\{v_1, v_2\}$ and $\{u_1, u_2, ..., u_N\}$ be the node sets of K_2 and Q_n, respectively, where $N = 2^n$ and $n \ge 1$. We assume that

both hypercubes K_2 and Q_n are G-labeled. That is, if G-label(u) denotes the Gray-code label of the node u, then we have:

(i) G-label(v_1) = 0,
(ii) G-label(v_2) = 1, and
(iii) G-label(u_i) = $b_{n-1}b_{n-2} \ldots b_1b_0$, where $b_{n-1}b_{n-2} \ldots b_1b_0$ be an n-bit string such that the Hamming distance between two nodes is 1 if and only if the nodes are adjacent in the hypercube Q_n.

Let X = $b_{n-1}b_{n-2} \ldots b_1b_0$ be a binary string of length n, where b_{n-1} is the most significant bit and b_0 is the least significant bit. Then, 0(X) (resp. 1(X)) denotes the binary string $b_nb_{n-1}b_{n-2} \ldots b_1b_0$ of length n+1, where b_n = 0 (resp. b_n = 1).

We have seen that $Q_{n+1} = K_2 \times Q_n$ and $V(Q_{n+1}) = \{x_1, x_2, \ldots, x_N, y_1, y_2, \ldots, y_N\}$, where $x_i = (v_1, u_i)$ and $y_i = (v_2, u_i)$, $1 \le i \le N$. We label the nodes of the hypercube Q_{n+1} as follows:

(i) label(x_i) = 0(X)
(ii) label(y_i) = 1(X)

where X = G-label(u_i), $1 \le i \le N$.

It is easy to see that the assignment of labels which is performed by the above hypercube-labeling method forms a Gray-code labeling of the hypercube Q_{n+1}. We call this hypercube-labeling method *GL-method* and a hypercube labeled by the GL-method *GL-labeled hypercube*.

Let $V(Q_{n+1}) = \{u_1, u_2, \ldots, u_{2N}\}$ be the node set of the hypercube Q_{n+1} and let H_n and F_n be two subgraphs of Q_{n+1} induced be the sets $V(H_n)$ and $V(F_n)$, where $V(H_n)$ (resp. $V(F_n)$) contains all the nodes u of Q_{n+1} whose the most significant bit of G-label(u) is 0 (resp. 1). It is easy to see that H_n and F_n are two n-dimensional hypercubes on $N = 2^n$ nodes. It is also easy to see that if we assign a new label to each node u of the hypercubes H_n and F_n by deleting the most significant bit of the G-label(u), then both hypercubes H_n and F_n are G-labeled. Hereafter, the hypercubes H_n and F_n will be referred to as *upper hypercube* and *lower hypercube* of the hypercube Q_{n+1}, respectively.

The main properties of the construction and G-labeling of a hypercube are summarized in the following lemma.

Lemma 2.3 Let Q_{n+1} be a GL-labeled hypercube with $N = 2^{n+1}$ nodes and let G-label(u) be the Gray-code label of the node u, where $u \in V(Q_{n+1})$. We partition the hypercube Q_{n+1} with respect to node s, where G-label(s) = 00...0, and let $N(s, 0)$, $N(s, 1)$, ..., $N(s, n+1)$ be the adjacency-level sets of Q_{n+1}. Then, the following are hold:

(i) $V(Q_{n+1}) = V(H_n) \cup V(F_n)$, where H_n and F_n are the upper and lower hypercubes of Q_{n+1};
(ii) The most significant bit of the G-label(u) of a node $u \in V(Q_{n+1})$ is 0 (resp. 1) if and only if $u \in V(H_n)$ (resp. $u \in V(F_n)$).
(iii) Every node v of the set $V(H_n)$ is joined by an edge with one and only one node u of the set $V(F_n)$ (see criterion C3). Moreover, if G-label(v) = 0(X), then G-label(u) = 1(X), where X is a binary string of length n.

Next, we present the main theorem of this paper which provides the justification for an algorithm which assigns G-labels to a hypercube Q_n.

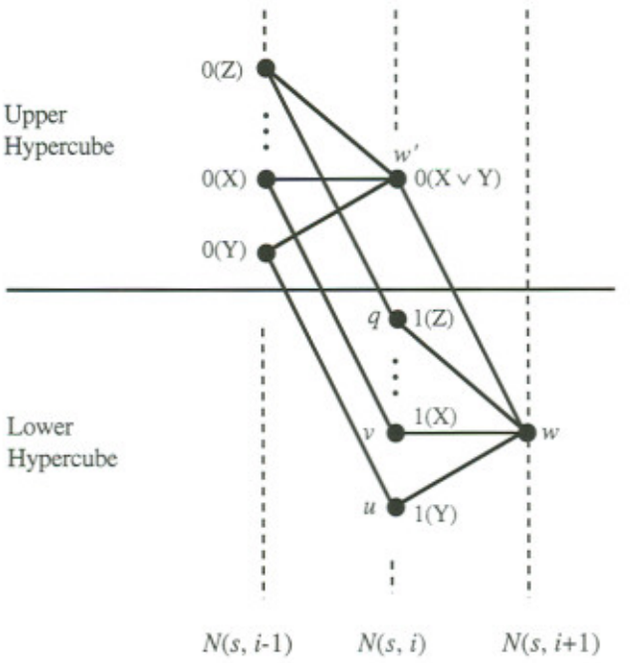

Fig. 3. Three consecutive adjacency-levels of the partition $\mathcal{L}(Q_n, s)$.

Theorem 2.1 Let $Q_n = (V_n, E_n)$ be an n-dimensional hypercube on $N = 2^n$ nodes, and let $N(s, 0)$, $N(s, 1)$, ..., $N(s, n)$ be the adjacency-level sets of the partition $\mathcal{L}(Q_n, s)$, where $s \in V_n$. Let u, v be distinct nodes in $N(s, i)$ and let w be a node in $N(s, i+1)$ such that $(w, u) \in E_n$ and $(w, v) \in E_n$, $1 \le i \le n-1$. Then, G-label(w) = G-label(u) $\vee$ G-label(v) is a Gray-code label of the node w.

Proof. We shall prove the theorem by induction on the dimension n of the hypercube Q_n. Let K_2 and Q_1 be two hypercubes and let $V(K_2) = \{s, u\}$ and $V(Q_1) = \{v, w\}$ be their node sets. Let $Q_2 = K_2 \times Q_1$. Thus, $V(Q_2) = \{s, u, v, w\}$ and let $\{s, u\}$ and $\{v, w\}$ be the node sets of the upper and lower hypercubes of Q_2, respectively. We assign the label 00 to the node s, that is, G-label(s) = 00. Then, G-label(u) = 01 and G-label(v) = 10. Obviously, G-label(w) = G-label(u) $\vee$ G-label(v) = 11 and $w \in N(s, 2)$. Therefore, in the base case $n = 2$ the induction hypothesis is hold.

Assume that the induction hypothesis holds for a hypercube Q_{n-1} on $N = 2^{n-1}$ nodes, $n \ge 2$. We shall show that it also holds for Q_n. Let $Q_n = (V_n, E_n)$ be an n-dimensional hypercube on $N = 2^n$ nodes, and let $N(s, 0)$, $N(s, 1)$, ..., $N(s, n)$ be the adjacency-level sets of the partition $\mathcal{L}(Q_n, s)$, where $s \in V_n$. Moreover, let H_n and F_n be the upper and lower hypercubes of Q_n. Then, H_n and F_n are two hypercubes each on $N = 2^{n-1}$ nodes.

Suppose that we have labeled the nodes of the levels $N(s, 0)$, $N(s, 1)$, ..., $N(s, i)$ of the partition $\mathcal{L}(Q_n, s)$, by using the given condition. Let w be a node in $N(s, i+1)$ and let $w \in V(F_n)$. Let w' be a node such that $w' \in V(F_n)$, $w' \in N(s, i)$, $(w', w) \in E_n$ and G-label(w') = 0(X$\vee$Y); See Fig. 3. We shall prove that G-label(w) = 1(X$\vee$Y).

It is easy to see that all the labeled neighbours of the node w' belong to the upper hypercube. Thus, by the induction hypothesis we have 0(X$\vee$Y) = ... = 0(X$\vee$Z) = ... = 0(Y$\vee$Z). We distinguish four cases:

Case I: label(w) = label(u) OR label(v) = (1$\vee$1)(X $\vee$ Y) = 1(X $\vee$ Y).
Case II: label(w) = label(u) OR label(q) = (1$\vee$1)(X $\vee$ Z) = 1(X $\vee$ Y).
Case III: label(w) = label(u) OR label(w') = (1$\vee$0)(X $\vee$ (X $\vee$ Y)) = 1(X $\vee$ Y).

Case IV: label(w) = label(q) OR label(w') $= (1 \vee 0)(Z \vee (X \vee Y)) = 1((Z \vee X) \vee Y)$
$= 1((X \vee Y) \vee Y) = 1(X \vee Y)$.

So indeed label(w) is a G-label of the node w of the hypercube Q_n. Thus, the theorem follows for all values of n. □

3. The Gray-code Labeling Algorithm

We now present an optimal algorithm for Gray-code labeling of the nodes of an n-dimensional hypercube. The basic idea of the algorithm is motivated by the characterizations provided by Theorems 2.1. The algorithm is very simple and uses breadth-first search and logical disjunction on n-bit strings; it visits the nodes of the hypercube by using the breadth-first search and computes the G-label of a node u by performing the logical disjunction of the G-labels of two nodes adjacent to node u. More precisely, the algorithm takes as input the adjacency list of an n-dimensional hypercube $Q_n = (V_n, E_n)$ with $N = 2^n$ nodes and operates as follows:

(1) Select an arbitrary node $s \in V_n$ and label node s and its adjacent nodes properly; obviously, this process computes the G-labels of all the nodes of the sets $N(s, 0)$ and $N(s, 1)$.
(2) Determine the next non G-labeled node u of the set $N(s, i)$ by using breadth-first search, $2 \le i \le \log N$.
(3) Select two arbitrary nodes a and b of the set $N(s, i\text{-}1)$ which are adjacent to u, and compute the G-label of u by performing the logical disjunction of G-label(a) and G-label(b).

From the above described Gray-code labeling method, it is clear that the G-label of a node u is the binary representation of G-label(a) $\vee$ G-label(b), where $\vee$ is the logical OR operation and a, b are two Gray-code labeled nodes which are adjacent to node u.

The nodes of the hypercube Q_n are visited in the order in which they are selected in stage (2) of the algorithm. This visiting process fixes the G-labels of the nodes of Q_n. According to the process of visiting the nodes of Q_n, the labeling of the nodes of the adjacency-level set $N(s, i)$ cannot start, unless all nodes of set $N(s, i\text{-}1)$ are labeled. Thus, the two nodes a and b, which are arbitrarily selected of the set $N(s, i\text{-}1)$ in stage (3), are always G-labeled. The correctness of the algorithm is established through the Theorem 2.1. Its proof relies on the results of Section 2 (see Lemmas 2.1, 2.2 and 2.3).

Having described the Gray-code labeling algorithm and proved its correctness, we are now in a position to estimate its time complexity. In doing so we give a more formal listing of the algorithm (see Fig. 4).

Let us first comment on some important features of the algorithm. For each node u of Q_n we maintain four variables: G-label(u), Color(u), A(u) and B(u). The variable Color(u) takes the colors White, Gray or Blank, and the variables A(u) and B(u) maintain the predecessors of the node u from the previous adjacency levels. To keep track of progress, first the algorithm colors each node White, then Gray and finally Blank (actually, the coloring process is due to breadth-first search). Initially, Color(u) is White for every nodes u of Q_n. After the computation of the G-label of a node w, it becomes Black. Consequently, the algorithm determines the node u which is adjacent to w and has White or Gray color. Then, it assigns the node w to A(u), in the case where A(u) = u (if a node $v \neq u$ has been assigned to A, then the algorithm assigns the node w to B(u), again in the case where B(u) = u). If u is a White node, then u is inserted into the queue Q (enqueue operation) and is

colored Gray. Later, the algorithm determines the Gray node u at the head of the queue Q, computes the G-label(u) of the node u by performing the logical disjunction of G-label(A(u)) and G-label(B(u)), delete u from the queue Q (dequeue operation) and colors it Black. Since, the algorithm inserts only White nodes into the queue Q and then colors them Gray, we easily conclude that each node of the hypercube is enqueued at most once, and hence dequeued at most once. For simplicity, the n-bit string consisting of all zeros will be denoted by 0^n.

Algorithm `G-labeling`;
begin

1. **for** each $u \in V_n$ **do** Color(u) $\leftarrow$ White; A(u) $\leftarrow$ B(u) $\leftarrow$ u;
2. choose an arbitrary node $s \in V_n$; G-label(s) $\leftarrow$ 0^n; Color(s) $\leftarrow$ Black;
3. G-label(u_k) $\leftarrow$ 0^n, for every node $u_k \in$ Adj(s), $0 \le k \le n$-1;
4. **for** each node $u_k \in$ Adj(s), $0 \le k \le n$-1, **do**
   - 4.1 update G-label(u_k) by toggling its k-th bit, and then assign Color(u_k) $\leftarrow$ Black;
   - 4.2 **for** each node $v \in$ Adj(u_k) **do**
     - **if** A(v) = v **then** A(v) $\leftarrow$ u_k
       elseif B(v) = v **then** B(v) $\leftarrow$ u_k;
     - **if** Color(v) = White **then** Enqueue(Q, v); Color(v) $\leftarrow$ Gray;
5. **while** Q $\neq \varnothing$ **do**
   - 5.1 $u \leftarrow$ head(Q);
   - 5.2 G-label(u) $\leftarrow$ G-label(A(u)) `OR` G-label(B(u));
   - 5.3 **for** each node $v \in$ Adj(u) **do**
     - **if** Color(v) $\neq$ Black **then**
       - **if** A(v) = v **then** A(v) $\leftarrow$ u
         elseif B(v) = v **then** B(v) $\leftarrow$ u;
     - **if** Color(v) = White **then** Enqueue(Q, v); Color(v) $\leftarrow$ Gray;
   - 5.4 Dequeue(Q); Color(u) $\leftarrow$ Black;

end;

Fig. 4. The Gray-code labeling algorithm for n-dimensional hypercubes.

We shall assume a generic one-processor, RAM (Random Access Machine) model of computation as our implementation technology. In this model, the operations of enqueuing and dequeuing take $O(1)$ time, while both logical `OR` operation $x \vee y$ and assignment operation $x \leftarrow y$ take $O(\log n)$ time in the case where x and y are binary n-strings.

Let us now analyze the computational complexity of the algorithm. We shall obtain its overall complexity by computing the complexity of each step separately. The complexity of the algorithm is analyzed as follows: *Step 1*. Clearly, this initialization step takes $O(\log N)$ time, where $N = 2^n$. *Step 2*. Since G-label is a binary n-string, the main operation of this step requires $O(\log N)$ time. *Step 3*. Based on the operation of the step 2, it follows that this step requires $O(\log^2 N)$ time. *Step 4*. The `for` loop of this step is executed k times, $0 \le k \le n$-1. Thus, the substeps 4.1 and 4.2 are executed exactly $\log N$ times. It is easy to see that substep 4.1 requires $O(\log N)$ time. The `for` loop of substep 4.2 is executed $O(\log N)$ time. In total, step 4 is executed in $O(\log^2 N)$ time. *Step 5*. The `while` loop of this step iterates as long as there remain White nodes in the hypercube. Since each node of the

hypercube is enqueued at most once, and hence dequeued at most once, the substeps 5.1 through 5.4 are executed exactly N times. It is easy to verify that substeps 5.1 and 5.4 take $O(1)$ time, while substeps 5.2 and 5.3 take $O(\log N)$ time. Thus, step 5 is executed in $O(N \log N)$ time.

Taking into consideration the time complexity of each step of the algorithm, we conclude that the total running time of the algorithm is bounded by $O(N \log N)$. Thus, the results of this section can be summarized in the following theorem.

Theorem 3.2 Given an n-dimensional hypercube Q_n on $N = 2^n$ nodes, the algorithm `G-labeling` correctly produces a Gray-code labeling of Q_n in $O(N \log N)$ time. In view of the fact that Q_n has n 2^{n-1} edges, this behaviour is optimal.

4. The Recognition Algorithm

In this section, we present an optimal algorithm for the problem of recognizing n-dimensional hypercubes. Our hypercube labeling algorithm `G-labeling` is used as a basis for the proposed recognition algorithm.

Let $\mathcal{L}(Q_n, s)$ be a partition of an n-dimensional hypercube $Q_n = (V_n, E_n)$, with respect to $s \in V_n$, and let $N(s, 0)$, $N(s, 1)$, ..., $N(s, n)$ be the adjacency-levels of $\mathcal{L}(Q_n, s)$. The following statements hold.

(P5) The adjacency-levels $N(s, 0)$, $N(s, 1)$, ..., $N(s, n)$ are independent sets (see Lemma 2.1).

(P6) The adjacency-level set $N(s, p)$ has $n! / (p! (n - p)!)$ nodes, $0 \le p \le n$.

(P7) Each node of the set $N(s, p)$ has p adjacent nodes in $N(s, p\text{-}1)$ and $n - p$ adjacent nodes in $N(s, p+1)$, $1 \le p \le n\text{-}1$.

Next, we show the relationship between the G-label of a node of the level $N(s, p)$ and its distance of the nodes of the level $N(s, 1)$, $2 \le p \le n$. We prove the following result.

Lemma 4.1 Let $Q_n = (V_n, E_n)$ be an n-dimensional hypercube on $N = 2^n$ nodes, and let $N(s, 0)$, $N(s, 1)$, ..., $N(s, n)$ be the adjacency-level sets of the partition $\mathcal{L}(Q_n, s)$, where $s \in V_n$. Let u and v_p be nodes of the sets $N(s, 1)$ and $N(s, p)$, respectively, $2 \le p \le n$, and let G-label$(u) = 0 \ldots 1 \ldots 0$, where 1 holds the i-th position, $0 \le i \le n\text{-}1$. Then, the G-label(v_p) has 1 in the i-th position if and only if $d(u, v_p) = p - 1$.

Proof. It follows directly from Theorem 2.1, since all the nodes of a path $[u, v_2, \ldots, v_p]$ have 1 in the i-th position of their G-labels, where $u \in N(s, 1)$ and $v_j \in N(s, k)$, $2 \le j \le p$ (see Fig. 2 or Fig.5a). □

Given an n-dimensional hypercube Q_n and the adjacency-level sets $N(s, 0)$, $N(s, 1)$, ..., $N(s, n)$ of the partition $\mathcal{L}(Q_n, s)$, we can easily show the following statements:

(P8) For every node v in the set $N(s, p)$, $2 \le p \le n$, there are exactly p nodes u in the set $N(s, 1)$ such that $d(u, v) = p - 1$, (see Criterion C2 and Property P6).

(P9) The G-label of every node v of the set $N(s, p)$ has exactly p 1's and, hence, $n\text{-}p$ 0's (see Lemma 4.1 and Theorem 2.1).

(P10) The G-labels of every pair of nodes u, v of the set $N(s, p)$ differ in exactly two bits if u, v have a common neighbour in the set $N(v, p\text{-}1)$, $1 \le p \le n\text{-}1$; otherwise, their G-labels differ in at least four bits, $n \ge 4$ (see Lemma 4.1 and Theorem 2.1).

We are now in a position to prove the following theorem which gives a necessary and sufficient condition for a given graph on $N = 2^n$ nodes to be an n-dimensional hypercube. Hereafter, we shall refer to the integer representation of the G-label of a node u as IG-label(u).

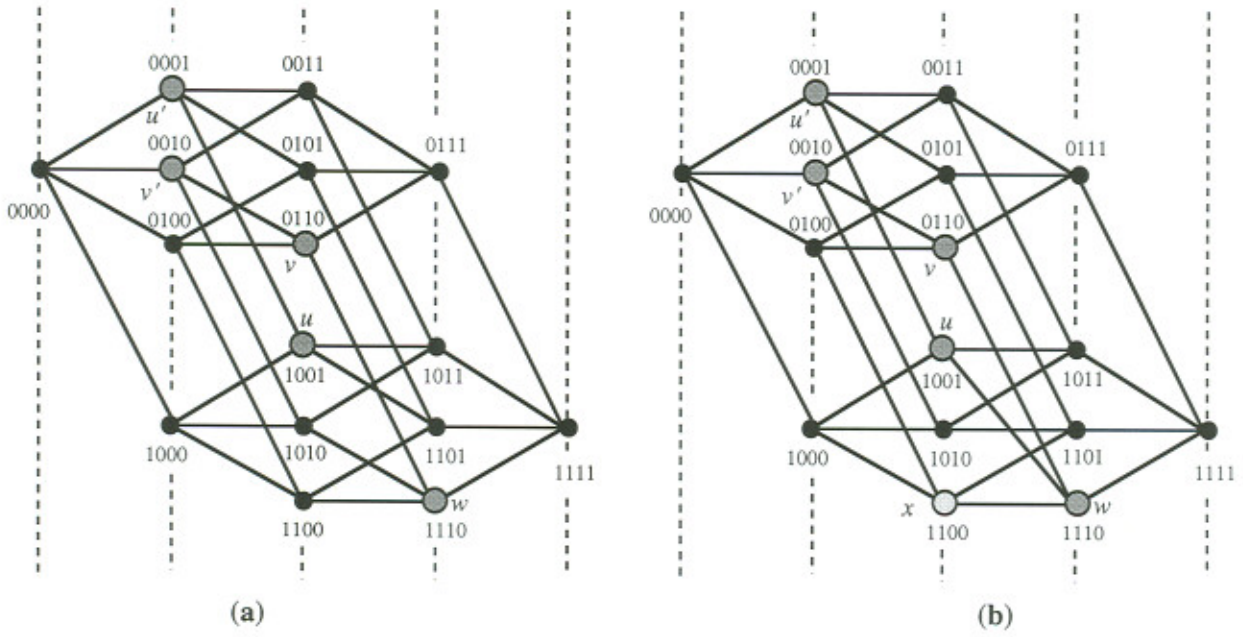

Fig. 5. (a) The hypercube Q_4; (b) A graph Q on $N = 2^4$ nodes; Q is not a hypercube.

Theorem 4.1 Let $Q = (V, E)$ be a graph on $N = 2^n$ nodes which satisfies the properties P5 - P7, and let IG-label(v_i) be the integer representation of the G-label of the node $v_i \in V$, $0 \le i \le N-1$. Then, Q is an n-dimensional hypercube if and only if for every $w \in V$ its adjacency nodes $u_0, u_1, ..., u_{n-1}$ can be ordered such that $|IG\text{-label}(w) - IG\text{-label}(u_k)| = 2^k$, $0 \le k \le n-1$.

Proof. ($\Rightarrow$) Since Q is a hypercube on $N = 2^n$ nodes, each node w in Q can be uniquely represented by an n-bit string using the algorithm `G-labeling`; that is, G-label(w). Thus, this implication follows directly from the properties P4 and P9 (see Fig. 5a).

($\Leftarrow$) Suppose now that Q is not a hypercube, and let $N(s, 0), N(s, 1), ..., N(s, n)$ be the adjacency-level sets of the partition $\mathcal{L}(Q, s)$, where $s \in V$. Since Q satisfies the properties P5 - P7, it follows that there exist a pair of adjacent edges which do not form a 4-cycle in Q. Therefore, there exist a node w in $N(s, p)$ and two neighbours u, v of w in $N(s, p-1)$ such that the pair of adjacent edges (u, w) and (w, v) do not form a 4-cycle in Q; that is, nodes u, v do not have a common neighbour in $N(s, p-2)$ (see Fig. 5b; $p = 3$). Suppose that $N(s, p)$ is the first adjacency-level set which contains such a node w. There are two cases to consider:

Case I: G-label(w) = G-label(v) $\vee$ G-label(x), where nodes v, x have a common neighbour in $N(s, p-2)$ and $x \neq u$. The G-labels of the nodes v, x differ in exactly two bits (see Property P10) and, thus, G-label(w) and G-label(v) differ in exactly one bit. Since u, v do not have a common neighbour in $N(s, p-2)$, it follows that G-label(u) and G-label(v) differ in at least four bits. Thus, G-label(v) and G-label(w) differ in more than one bit.

Case II: G-label(w) = G-label(u) $\vee$ G-label(v). Since G-label(u) and G-label(v) differ in at least four bits, it follows that G-label(u) and G-label(w) differ in more than one bit.

In both cases, we conclude that there exists a neighbour u of w such that G-label(u) and G-label(w) differ in more than one bit and, thus, $|IG\text{-label}(w) - IG\text{-label}(u)| \neq 2^k$, $0 \leq k \leq n-1$. □

Based on the conditions provided by Theorem 4.1, we next develop a recognition algorithm for Hypercubes. Obviously, these conditions contain enough information to enable the logical deduction that a given graph $Q = (V, E)$ on $N = 2^n$ nodes is indeed an n-dimensional hypercube. We call the recognition algorithm `G-recognition` and give its formal listing in Fig. 6.

Algorithm `G-recognition`;
begin
1. G-label the input graph $Q = (V, E)$ using the algorithm `G-labeling`;
2. if one of the properties P5 - P7 is not satisfied then Q is not a hypercube; exit;
3. Compute the integer-label IG-label(v), for each node $v \in V$;
4. Initialize the Boolean array $D[1..N] \leftarrow F$;
5. **for** each node $w \in V$ **do**
   - 5.1 **for** each node $u \in$ Adj(w) **do**
     compute $k = |IG\text{-label}(w) - IG\text{-label}(u)|$;
     set $D[k] \leftarrow T$;
   - 5.2 **for** $k = 2^i$, $i = 0, 1, 2, ..., n-1$, **do**
     if $D[k] = F$ then Q is not a hypercube; exit;
   - 5.2 **for** $k = 2^i$, $i = 0, 1, 2, ..., n-1$, **do** set $D[k] \leftarrow F$;
6. Return: Q is a hypercube;

end;

Fig. 6. The recognition algorithm for n-dimensional hypercubes.

Let us now analyze the computational complexity of the recognition algorithm. We shall follow a step-by-step analysis. *Step 1*. We have proved that the time complexity of the algorithm `G-labeling` is $O(N \log N)$; see Theorem 3.2. Moreover, we have seen that this algorithm computes the adjacency-level sets of the input graph. Within the recognition algorithm, we can easily modify the labeling algorithm `G-labeling` so that it computes the adjacency-levels of any graph Q in $O(N \log N)$ time; the algorithm terminates in the case where Q is not an n-dimensional hypercube. *Step 2*. Since an n-dimensional hypercube has N nodes and $N \log N$ edges, it is easy to see that for any graph Q we can decide whether the statements P5, P6 and P7 hold in $O(N \log N)$ time. *Step 3*. Given the G-labels of the nodes of Q, we can easily compute the corresponding integer labels of Q in $O(N \log N)$ time. *Step 4*. Obviously, it takes $O(N)$ time. *Step 5*. The `for` loop of this step is executed for every node $w \in V$. Each of the `for` loops of the substeps 5.1, 5.2 and 5.3 is executed exactly $\log N$ times. Thus, the total computational time of the Step 5 is $O(N \log N)$. *Step 6*. This step returns the message "Q is a hypercube"; hence, it takes $O(1)$ time.

Taking into consideration the time complexity of each step of the algorithm, we conclude that the algorithm runs in $O(N \log N)$ time being, therefore, optimal. Thus, we have proved the following result.

Theorem 4.2 Given a graph Q with $N = 2^n$ nodes and $n2^{n-1}$ edges, the algorithm `G-recognition` recognizes whether Q is an n-dimensional hypercube in $O(N \log N)$ time. In view of the fact that Q has $n2^{n-1}$ edges, this behaviour is optimal.

5. Parallel Implementation

In this section, we present optimal parallel Gray-code labeling and recognition algorithms for n-dimensional hypercubes $Q_n = (V_n, E_n)$. The first labeling algorithm is a straightforward parallel implementation of the sequential algorithm `G-labeling` described in Section 3. The basic idea of the second labeling algorithm is motivated by the characterizations provided by Lemma 4.1; that is, the algorithm is based on certain properties of the Gray-code labeling and the adjacency-level sets of a Q_n. This algorithm is executed in $O(1)$ time and it is optimal in the case where the adjacency-level sets are given.

In order to design and analyze our parallel algorithms, we must choose an appropriate model for parallel computation. In this work, we shall use the PRAM (Parallel Random Access Machine) model as a formal computation model. Our notion of optimality is based on the Work-Time (WT) framework [9]. The *work* is defined to be the total number of operations used by a parallel algorithm (it has nothing to do with the number of processors available), while the *time* is defined to be the total number of time units required by a parallel algorithm (during each time unit a number of concurrent operations can take place). Within the WT framework, a parallel algorithm for solving a given computational problem will be called *optimal* if the work $W(n)$ required by the algorithm is asymptotically the same as the sequential complexity of the problem, regardless of the running time $T(n)$ of the parallel algorithm.

Our parallel algorithms in this section are given in the WT presentation and our treatment of the optimality notion is so far concentrated on the WT framework.

5.1 Parallel labeling algorithm A

We first describe a parallel implementation of the optimal sequential algorithm `G-labeling`. The details are spelled out by the algorithm `PG-labeling_A` which we present next.

Algorithm `PG-labeling_A`;
begin

1. **for** each $u \in V_n$ **do in parallel** Color(u) $\leftarrow$ White;
2. choose an arbitrary node $s \in V_n$; G-label(s) $\leftarrow 0^n$; Color(s) $\leftarrow$ Black;
3. **for** every node $u_i \in$ Adj(s), $0 \le i \le n$-1, **do in parallel**
   - 3.1 G-label(u_i) $\leftarrow$ 0 ... 1 ... 0; (update the label of u_i by toggling its i-th bit)
   - 3.2 Color(u_i) $\leftarrow$ Black;
   - 3.3 **for** every node $v_j \in$ Adj(u_i), $0 \le j \le n$-1, **do in parallel**
     Color(v_j) $\leftarrow$ Gray;
4. **for** $i = 2, 3, ..., n$, **do**
   - 4.1 **for** every node $u \in V_n$ such that Color(u) = Gray **do in parallel**
     - 4.1.1 choose two arbitrary nodes x and y from the set Adj(u) such that Color(x) = Color(y) = Black;
     - 4.1.2 G-label(u) $\leftarrow$ G-label(x) `OR` G-label(y); Color(u) $\leftarrow$ Black;
     - 4.1.3 **for** every node $v \in$ Adj(u) such that Color(u) = White **do in parallel**
       Color(v) $\leftarrow$ Gray;

end;

The correctness of the algorithm `PG-labeling_A` is established through the correctness of the

sequential algorithm `G-labeling`. Next, we analyze the computational complexity of the algorithm on a CRCW PRAM [9, 15].

We follow a step-by-step analysis. *Step 1*. Clearly, this initialization step can be executed in $O(1)$ time, using $O(N)$ operations, where $N = 2^n$. *Step 2*. This step can be completed in $O(1)$ time, using $O(\log N)$ operations, since the length of the binary n-string G-label is $n = \log N$. *Step 3*. The `for` loop considers each node u in the adjacency set Adj(s) which contains $\log N$ nodes. It is easy to see that, the substeps 3.1 and 3.2 are executed in $O(1)$ time and use $O(\log N)$ and $O(1)$ operations, respectively. The `for` loop of substep 3.3 is executed in $O(1)$ parallel time using $O(\log N)$ operations. Thus, overall, step 3 is executed in $O(1)$ time using $O(\log^2 N)$ operations. *Step 4*. The `for` loop of this step is executed $O(\log N)$ times. In the i-th iteration of this loop, the Gray nodes selected in substep 4.1 are exactly $k_i = |\, N(s, i)\,|$, $i = 2, 3, ..., n$. It is easy to verify that, the body of the loop of substep 4.1 can be executed in $O(1)$ time using $O(\log N)$ operations. Thus, step 4 is executed in $O(\log N)$ time using $k_2 \log N + k_3 \log N + ... + k_n \log N$ operations. Since, $k_2 + k_3 + ... + k_n = N - (\log N + 1)$, the total number of operations used by step 4 is $O(N \log N)$.

Taking into consideration the time complexity of each step of the algorithm, as well as the number of operations used by each step, we can present the following result.

Theorem 5.1 Given an n-dimensional hypercube Q_n with $N = 2^n$ nodes, the parallel algorithm `PG-labeling_A` correctly produces a Gray-code labeling of the nodes of Q_n; furthermore, it can be executed in $O(\log N)$ time using a total of $O(N \log N)$ operations on a CRCW PRAM.

5.2 Parallel labeling algorithm B

In this section we describe a parallel hypercube-labeling algorithm which is mainly based on the relationship between the G-label of a node of level $N(s, p)$ and its distance of the nodes of the level $N(s, 1)$ of a partition $\mathcal{L}(Q_n, s)$, where $s \in V_n$.

Recall that the G-label of a node v of the set $N(s, p)$ has 1 in the i-th position if and only if there exists a node u in the set $N(s, 1)$ such that $d(u, v) = p - 1$ and G-label(u) has 1 in the i-th position, $2 \le p \le n$; see Theorem 4.1. Based on this result we proceed to formulate a parallel algorithm for Gray-code labeling an n-dimensional hypercube $Q_n = (V_n, E_n)$. The details of this algorithm, which we call `PG-labeling_B`, are spelled out as follows.

Algorithm `PG-labeling_B`;
begin

1. Let $N(u, 0), N(u, 1), ..., N(u, n)$ be the adjacency-level sets of the partition $\mathcal{L}(Q_n, u)$, for every $u \in \{s\} \cup \text{Adj}(s)$, where $s \in V_n$;
2. Set G-label(s) $\leftarrow 0^n$ and G-label(u_k) $\leftarrow 0^n$ for every $u_k \in N(s, 1)$;
3. **for** every node $u_k \in N(s, 1)$ **do in parallel**
   G-label(u_k) $\leftarrow 0 \ldots 1 \ldots 0$; (update the label of u_k by toggling its k-th bit)
4. **for** every node $v \in V - \{N(s, 0) \cup N(s, 1)\}$ **do in parallel**
   4.1 **for** every node $u_k \in N(s, 1)$ **do in parallel**
   if $d(v, s) = d(v, u_k) + 1$ then update the G-label(v) by toggling its k-th bit;

end;

Let us now analyze the computational complexity of algorithm `PG-labeling_B`. Here, as a model of parallel computation we shall use a CREW PRAM [9, 15].

The complexity of the algorithm is analyzed in a step-by-step fashion as follows: *Step 1*. Suppose that the adjacency-level sets $N(u, 0)$, $N(u, 1)$, ..., $N(u, n)$ are given for every $u \in \{s\} \cup \text{Adj}(s)$, where $s \in V_n$. *Step 2*. Obviously, this step takes $O(1)$ time using $O(\log N)$ operations. *Step 3*. The adjacency-level set $N(s, 1)$ contains $\log N$ nodes and, thus, this step is executed in $O(1)$ time with $O(N \log N)$ operations. *Step 4*. The number of nodes in the sets $V - \{N(s, 0) \cup N(s, 1)\}$ and $N(s, 1)$ are $N - (\log N+1)$ and $\log N$, respectively. The operation of testing whether or not a node updates the i-th bit of its G-label can be executed in $O(1)$ time with $O(1)$ operations. Thus, this step is completed in $O(1)$ time using $O(N \log N)$ operations.

From the above step-by-step analysis, we obtain the overall computational complexity of the algorithm. Thus, the following theorem holds.

Theorem 5.2 Let $Q_n = (V_n, E_n)$ be an n-dimensional hypercube with $N = 2^n$ nodes. The parallel algorithm `PG-labeling_B` correctly produces a Gray-code labeling of the nodes of Q_n and runs in $O(1)$ time using in total of $O(N \log N)$ operations on a CREW PRAM model provided we have the adjacency-levels of a partition $\hat{L}(Q_n, u)$, for every $u \in \{s\} \cup \text{Adj}(s)$, where $s \in V_n$.

Remark 5.1 Following the work-time analysis of the parallel algorithms presented in this section, we are led to the conclusion that the Gray-code labeling of a hypercube can be optimally computed (in view of the property P2) in a parallel process environment. We should point out that, the adjacency-level sets $N(u, 0)$, $N(u, 1)$, ..., $N(u, n)$ can be computed in $O(\log N)$ time using $O(N \log^2 N)$ operations, where $u \in \{s\} \cup \text{Adj}(s)$ and $s \in V_n$. They also can be computed using the information contained in the u-th row of the $N \times N$ distance matrix D of Q_n. Thus, the adjacency-levels of the partition $\hat{L}(Q_n, u)$, for every $u \in V_n$, can be computed in $O(\log\log N)$ time using $O(M(N))$ operations on the CREW PRAM model, where $M(N)$ is the best known sequential bound for multiplying two $N \times N$ matrices; see Section 2 or [9].

5.3 Parallel recognition of hypercubes

In this section we show that the problem of recognizing whether a given graph Q on $N = 2^n$ nodes is indeed an n-dimensional hypercube can be solved in parallel. We present an optimal parallel recognition algorithm which is based on the conditions given by Theorem 4.1; that is, our algorithm is a straightforward parallel implementation of the sequential algorithm `G-recognition` described in Section 4.

Obviously, the step 1 of the algorithm `G-recognition` can be implemented using the parallel algorithms `PG-labeling_A` and `PG-labeling_B`. Moreover, we can easily show that the steps 3 through 6 of the same algorithm can be implemented in $O(\log N)$ time using in total of $O(N \log N)$ operations on a CREW PRAM model.

Let us now focus on the parallel implementation of the step 2 of the sequential algorithm `G-recognition`. Given the adjacency-levels $N(s, 0)$, $N(s, 1)$, ..., $N(s, n)$ of the partition $\hat{L}(Q, s)$, we can decided whether these sets are independent in $O(1)$ time using in total of $O(N \log N)$ operations on a CRCW PRAM model. Obviously, we can count the elements of the sets $N(s, 0)$, $N(s, 1)$, ..., $N(s, n)$ in $O(\log N)$ time using $O(N)$ operations on a EREW PRAM model. Moreover, we can test whether a node u of the set $N(s, p)$ has p adjacent nodes in the set $N(s, p\text{-}1)$ and $n - p$ adjacent nodes in the set $N(s, p+1)$ in $O(\log N)$ time using $O(N \log N)$ operations on a CREW PRAM model, $1 \le p \le n\text{-}1$.

Taking into consideration the above analysis, as well as the computational complexity of the parallel labeling algorithms, we conclude that we can recognize whether a given graph Q is a

hypercube in $O(\log N)$ time using $O(N \log N)$ operations on a CRCW PRAM model. Thus, the results of this section can be summarized in the following theorem.

Theorem 5.3 Let $Q = (V, E)$ be a graph on $N = 2^n$ nodes. The problem of recognizing whether Q is an n-dimensional hypercube can be solved in $O(\log N)$ time using in total of $O(N \log N)$ operations on a CRCW PRAM model of computation.

6. Conclusions

In this paper we presented an optimal greedy algorithm for Gray-code labeling and recognizing n-dimensional hypercubes. Our labeling algorithm uses breadth-first search to guide the greedy chose of labels and computes the Gray-code label of a node u by performing the logical disjunction of the Gray-code labels of two nodes adjacent to node u. The algorithm is very simple and runs in $O(N \log N)$ time, where $N = 2^n$ is the number of nodes in the hypercube.

The idea of our algorithm is motivated by the work performed by Bhagavathi *et. al.* [2]. They presented an optimal algorithm which takes as input an n-dimensional hypercube Q_n and returns a Gray-code labeling of the nodes of the hypercube Q_n. The main feature of their algorithm is that it visits the nodes of the hypercube by using the depth-first search. Based on the labeling algorithm we proposed an optimal recognition algorithm for hypercubes; that is, our algorithm recognizes whether a given graph on $N = 2^n$ nodes is indeed an n-dimensional hypercube and runs in $O(N \log N)$ time.

It is well-known that the depth-first search is a hardly parallelisable problem; it is a *P-complete* problem. On the other hand, it is also well-known that the breadth-first search can be efficiently implemented in parallel using matrix multiplication. Based on these facts, as well as on the properties of the GL-method, we proposed two parallel algorithms for the Gray-code labeling problem. The first algorithm is a straightforward implementation of our sequential algorithm and $O(\log N)$ time using a total of $O(N \log N)$ operations on a CRCW PRAM model of computation. The second algorithm takes advantage of certain topological properties of a hypercube Q_n and runs in $O(1)$ time using $O(N \log N)$ operations on a CREW PRAM, in the case where the adjacency-levels of a partition $\mathcal{L}(Q_n, u)$ are given, for every $u \in \{s\} \cup \mathrm{Adj}(s)$, where $s \in V_n$. We also show that a graph on $N = 2^n$ nodes can be recognized whether it is an n-dimensional hypercube in $O(\log N)$ time using in total of $O(N \log N)$ operations on a CRCW PRAM model of computation.

References

[1] D.P. Bertsekas and J.N. Tsitsiklis, *Parallel and Distributed Computations: Numerical Computations*, Prentic-Hall, Englewood Cliffs, NJ, 1989.

[2] D. Bhagavathi, C.E. Grosch and S. Olariu, A greedy hypercube-labeling algorithm, *The Computer Journal* **37** (1994) 124-128.

[3] A. Coppesmith and S. Winograd, Matrix Multiplication via Arithmetic Progression, in *STOC'87*. Also in *J. of Symbolic Computations* **9** (1990) 251-280.

[4] T. Cormen, C. Leiserson and R. Rivest, *Introduction to Algorithms*, MIT Press - McGraw Hill, 1991.

[5] S. Foldes, A characterization of hypercubes, *Discrete Math.* **17** (1977) 155-159.

[6] M.R. Garey and R.L. Graham, On cubical graphs, *J. Combin. Theory B* **16** (1975) 84-95.

[7] F. Harary, *Graph Theory*, Addison-Wesley, Reading (1969).

[8] F. Harary and J. Nieminen, Convexity in graphs, *J. Differential Geometry* **16** (1981) 185-190.

[9] J. JáJá, *An Introduction to Parallel Algorithms*, Addison-Wesley Publishing Company, Inc. (1992).

[10] S.L. Johnsson and C.-T. Ho, Optimal broadcasting and personalized communications in hypercube, *IEEE Trans. Comput.* **C-38** (1989) 1249-1268.

[11] V. Kumar, A. Grama, A. Gupta and G. Karypis, *Introduction to Parallel Computing: Design and Analysis of Algorithms*, The Benjamin/Cummings Publishing Company, Inc. (1994).

[12] J.M. Laborde and S.P.R. Hebbare, Another characterization of hypercubes, *Discrete Math.* **39** (1982) 161-166.

[13] S. Lakshmivarahan and S.K. Dhall, *Analysis and Design of Parallel Algorithms*, McGraw-Hill Publishing Company (1990).

[14] S.D. Nikolopoulos and S.D. Danielopoulos, Parallel computation of perfect elimination schemes using partition techniques on triangulated graphs, *Computers and Math. with Applications* **29** (1995) 47-57.

[15] J. Reif (editor), *Synthesis of Parallel Algorithms*, Morgan Kaufmann Publishers, Inc., San Mateo, California (1993).

[16] Y. Saad and M.H. Schultz, Topological properties of hypercube, *IEEE Trans. Comput.* **C-37** (1988) 867-872.

[17] D.S. Scott and J. Brandenburg, Minimal mesh embedding in binary hypercubes, *IEEE Trans. Comput.* **C-37** (1988) 1284-1288.

[18] A.Y. Wu, Embeddings of tree networks into hypercubes, *J. Parallel Distr. Comput.* **2** (1985) 238-249.