Applied Mathematics and Computation 259 (2015) 620–627

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

# Paths and cycles identifying vertices in twisted cubes ☆

Pao-Lien Lai

*Department of Computer Science and Information Engineering, National Dong Hwa University, Shoufeng, Hualien 97401, Taiwan, ROC*

ARTICLE INFO

*Keywords:*
Identify
Fault diagnosis
Gray code
Twisted cubes
Path
Cycle

ABSTRACT

The hypercube is one of the most popular interconnection networks since it has simple structure and is easy to implement. The twisted cube is an important variation of the hypercube and preserves many of its desirable properties. Karpovsky et al. introduced the concept of identifying codes to model fault-detection in multiprocessor systems and Honkala et al. developed an identifying code by using cycles to identify the faulty processors in the hypercube. In this paper, we study the vertex identification problem on the twisted cube. We first propose an interesting construction scheme to build paths and cycles, and furthermore apply a minimum number of paths and cycles to identify the faulty processors of the twisted cube.



## 1. Introduction

As the need for high-speed parallel processing systems increases, so does the importance of ensuring the reliability of the processors in those systems. In order to maintain the reliability of a system (or network), efficient fault-diagnosis algorithms are desired to test the system and locate faulty processors. The concept of identifying code was first introduced in [24]. An illustration comes from fault diagnosis [27,32,33] in multiprocessor systems.

Preparata et al. [33] first introduced a model, the so called PMC model, for system level diagnosis in multiprocessor systems. In this model, it is assumed that a processor can test the faulty or fault-free status of another processor. Every processor performs tests on its neighbors based on the communication links between them. When one processor tests another, the tester conducts the tested processor to be fault-free or faulty depending on the test response; the result is always accurate if the tester is fault-free, but if the tester is faulty, it can output any test result, regardless of the status of the tested processor.

A lot of work has been done [3,4,6–8,12,13,21,25,29,34] for the identification of malfunctioning processors by using balls in the following way: selecting some of the processors (constituting the code), and each of them checks its $r$-neighborhood, i.e., all the processors that are within graphic distance $r$. The processor sends an alarm if it detects a fault in its neighborhood. The goal is to determine the exact location of the malfunctioning processor or that all the processors are fine based on these responses.

Another interesting idea of using paths and cycles to identify the processors instead of balls is first mentioned in [20,36] and implemented in hypercube. The mathematical problem can then be formulated as follows. To send test messages which they can be routed through this network in any way we like. What is the smallest number of messages we have to send if based on which messages safely come back we can declare which vertex is faulty (if any)? We call these the vertex identification problem.

☆ This work was supported in part by the National Science Council of the Republic of China under Contract NSC 101-2115-M-259-007 and NSC 102-2115-M-259-007.

*E-mail address:* baolein@mail.ndhu.edu.tw

http://dx.doi.org/10.1016/j.amc.2015.02.090
0096-3003/

**Table 1**
The relation between vertex $u \in V(TQ_3)$ and $C_i,\ 0 \leqslant i \leqslant 3$.

| $u$ | $C_3$ | $C_2$ | $C_1$ | $C_0$ |
|---|---|---|---|---|
| 000 | 1 | 1 | 1 | 1 |
| 001 | 1 | 1 | 1 | 0 |
| 010 | 1 | 1 | 0 | 1 |
| 011 | 1 | 1 | 0 | 0 |
| 100 | 1 | 0 | 1 | 1 |
| 101 | 1 | 0 | 1 | 0 |
| 110 | 1 | 0 | 0 | 1 |
| 111 | 1 | 0 | 0 | 0 |
| Note: | 1 if $u \in C_i$.<br>0 if $u \notin C_i$. | | | |

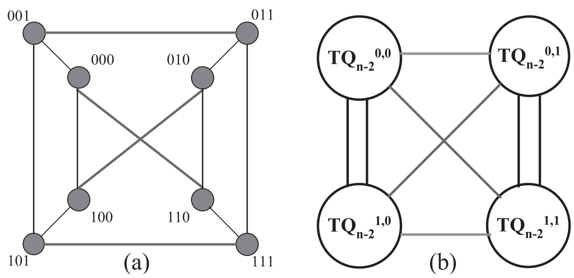


**Fig. 1.** Two illustrations of $TQ_3$ and $TQ_n$.

The classical binary reflected Gray code lists $n$-bit binary numbers so that successive numbers differ in exactly one bit position. Chen and Shin [9], Chen et al. [10,11], Lai et al. [28], Lai and Tasi [26], Li et al. [30] and Zheng et al. [38] have adopted Gray codes to build path and cycle skeletons, and furthermore applied paths and cycles on the process allocation, Hamiltonian problems, and torus embedding for hypercubes, crossed cubes, twisted cubes, and locally twisted cubes [11,22,23,31].

An $n$-dimensional twisted cube, $TQ_n$ [1,19], is an important variation of hypercube $Q_n$ and preserves many of its desirable properties [14–18,35]. $TQ_n$ has $2^n$ vertices and $n2^{n-1}$ edges, same as hypercube $Q_n$. However, the diameter, wide diameter, and faulty diameter in twisted cubes are about half of those in comparable hypercubes [5]. In this paper, we study the vertex identification problem on the twisted cube. We first apply the concept of Gray codes to propose some interesting construction schemes to build paths and cycles of a $TQ_n$. Furthermore, we give a minimum number of paths and cycles to identify the faulty vertices of a $TQ_n$.

The rest of this paper is organized as follows. The preliminary knowledge and fundamental definitions for twisted cubes and reflected link label sequence are given in the next section. Then, some properties are introduced for constructing paths and cycles in Section 3. Section 4 addresses how to identify vertices in twisted cubes with paths and cycles. Conclusions are given in the final section.

## 2. Preliminary

Network topology is usually represented by a graph where vertices represent processors and edges represent links between processors. In this paper, a network is represented as a finite and simple (i.e., without loops and multiple edges) undirected graph. For the graph definition and notation we follow [2]. $G = (V, E)$ is a graph if $V$ is a finite set and $E$ is a subset of $\{(u, v)|(u, v)$ is an unordered pair of $V\}$. We say that $V$ is the *vertex set* and $E$ is the *edge set*. Two vertices $u$ and $v$ are *adjacent* if $(u, v) \in E$. A *subgraph* of $G = (V, E)$ is a graph $(V', E')$ such that $V' \subseteq V$ and $E' \subseteq E$. Two subgraphs of $G$ are *vertex-disjoint* (or disjoint for short) if they have no common vertex. An *induced subgraph* is an edge-preserving subgraph, that is, $(V', E')$ is an induced subgraph of $(V, E)$ if and only if $V' \subseteq V$ and $E' = \{(u, v) \in E | u, v \in V'\}$.

A *walk* is a finite sequence of adjacent vertices denoted by $\omega = \langle \lambda_0, \lambda_1, \lambda_2, \ldots, \lambda_m \rangle$, for $0 \leqslant i \leqslant m$. In particular, a walk $\omega$ is called a *path* if all vertices $\lambda_0, \lambda_1, \lambda_2, \ldots, \lambda_m$ are distinct, with the possible exception $\lambda_0 = \lambda_m$. A *cycle* is a path with at least three vertices such that the first vertex is the same as the last one.

Let $u$ be a vertex of a $TQ_n$ and $u$ will be labeled as $u_{n-1}u_{n-2}\cdots u_1u_0$, a binary string of length $n$, where $u_{n-1}$ is the most significant bit and $u_0$ is the least significant bit. The $i$-bit (or bit-$i$) of $u$ is $u_i$ for $0 \leqslant i \leqslant n-1$. The complement of $u_i$ will be denoted by $\overline{u}_i$ ($\overline{0} = 1$ and $\overline{1} = 0$). To define twisted cube $TQ_n$, a parity function $f_i(u)$ is defined as $f_0(u) = u_0$ and $f_i(u) = u_i \bigoplus u_{i-1} \bigoplus \cdots \bigoplus u_1 \bigoplus u_0,\ 1 \leqslant i \leqslant n-1$, where $\bigoplus$ is the exclusive-or operation.

$TQ_1$ is a complete graph with two vertices labeled by 0 and 1, respectively. In the following, let $n \geqslant 3$ be an odd integer and we give the recursive definition of $n$-dimensional twisted cube $TQ_n$. The vertices of a $TQ_n$ can be decomposed into four