Correspondence

UPP Graphs and UMFA Networks—Architecture for Parallel Systems

David Goldfeld and Tuvi Etzion

Abstract—**A graph with unique path property (UPP) has 2^n vertices and a unique path of length n between each two vertices. These graphs are very important in the design of architectures and algorithms for interconnection networks. Given two UPP graphs we introduce an algorithm to determine isomorphism between the graphs. We give a construction for nonisomorphic UPP graphs and show a lower bound of $2^{(2^{n+1}-3n-2)/9}$ nonisomorphic graphs with 2^n vertices. UMFA (Uniform Minimal Full Access) networks are multistage interconnection networks, that use the same interconnection pattern between each two consecutive stages. These networks are derived from UPP graphs, and the most known network of this type is the Omega network. We discuss the problem of rearrangeability of UMFA networks.**

Index Terms—**BFS trees, buddy property, exchange operation, independent set, isomorphism, rearrangeability, UMFA network, UPP graph.**

I. Introduction and Definitions

Multistage Interconnection (MI) networks are a very important model in parallel processing. The most popular networks, the Omega network [1], the Flip network [2], the Baseline network [3], the Reverse Baseline network [3], the Indirect Binary cube [4], and the Modified Data Manipulator [5], are all topologically equivalent as proved by Wu and Feng [3], by Kruskal and Snir [6], and by Bermond, Fourneau, and Jean-Marie [7].

The Banyan networks were defined by Goke and Lipovski [8]. In a Banyan network there is a unique path from each input to each output, but there are no limitations on the degrees of the vertices, and no limitations on the lengths of the paths.

If a Banyan network has 2×2 switching elements (SE's) and the same number of SE's in each stage then it is called an MFA (Minimal Full Access) network [9]. When the interconnection pattern between successive stages is the same, the MFA network is called a UMFA (Uniform MFA) network. The Omega network and the Flip network are UMFA networks. Each UMFA network can be derived from a UPP (Unique Path Property) digraph. A digraph is a UPP graph of order n if it has $N = 2^n$ vertices with out-degree 2 and in-degree 2, and between any pair of vertices there is a unique path of length n [10]. We will assume throughout this paper that the vertices are labeled by $0, 1, \cdots, N-1$.

If we define the *distance*, $d_G(x, y)$ (or $d(x, y)$ if G is understood) between the vertices x and y as the shortest path from x to y we have that in a UPP graph G, $d_G(x, y) \leq n$ for every x and y.

The de Bruijn graph [11] of order n, G_n, is the most famous UPP graph. It has N vertices each one of which is represented by a different binary n-tuple. From vertex $X = (x_1, x_2, \cdots, x_n)$ to vertex $Y = (y_1, y_2, \cdots, y_n)$ there is a directed edge iff

$$y_i = x_{i+1}, \quad i = 1, 2, \cdots, n-1, \quad \text{and} \quad y_n \in \{0, 1\}.$$

Manuscript received August 31, 1988; revised November 4, 1991. T. Etzion was supported in part by Technion V.P.R. Fund–Elron-Elbit Electronics Research Fund.

The authors are with Computer Science Department, Technion–Israel Institute of Technology, Haifa 32000, Israel.

IEEE Log Number 9200188.

A path of length k in G_n can be represented by the sequence $x_1 x_2 \cdots x_{n+k}$, where $x_i x_{i+1} \cdots x_{i+n-1} \rightarrow x_{i+1} \cdots x_{i+n-1} x_{i+n}$, $1 \leq i \leq k$, is the ith edge in the path. From the de Bruijn graph we can derive one of the most popular networks, the Shuffle-Exchange network. An n-stage Shuffle-Exchange network is the Omega network.

Generally, given a UPP graph G we construct a UPP network NE as follows. The UPP network has the same N processors as the vertices in G. Let $p_0, p_1, \cdots, p_{N-1}$ be the permutation on the vertices, such that there is a directed edge from vertex i to vertex p_i in G. The operation in which processor i sends its information to processor p_i will be called the TRANSFER operation (SHUFFLE in the Shuffle-Exchange network). In a second operation called EXCHANGE, processor k can send its information to processor q_k such that $k \neq q_k$ and the vertices k and q_k have a common father in G. Thus, in the EXCHANGE operation a few processors take part and decide whether or not to perform the EXCHANGE, independent of the other processors. (In the Shuffle-Exchange network the EXCHANGE is performed by pairs of processors.) If the EXCHANGE operation is performed between more than two processors we have a cycle in which the information is transferred, by an arbitrary direction given to the cycle. A TRANSFER operation will be followed by an EXCHANGE operation and we will refer to them as a pass.

The *reverse*, G^R, of a digraph G, has the same underlying graph as G. In G^R, there is a directed edge from i to j, if in G there is a directed edge from j to i. The TRANSFER defined by the network which corresponds to the reverse graph is the REVERSE operation. (In the Shuffle-Exchange network this is the UNSHUFFLE.) Note, that if the graph has the buddy property, the effect of a TRANSFER followed by an EXCHANGE is the same as an EXCHANGE (which corresponds to the reverse graph) followed by a REVERSE.

An *alternating cycle* in a directed graph is an undirected cycle of even length, such that each two consecutive edges in the cycle are in opposite directions. A graph in which all the alternating cycles have length four has the buddy property [12]. In this case in the UPP network each EXCHANGE will be performed between two processors.

In Section II we introduce an algorithm that accepts as an input two UPP graphs and determines whether they are isomorphic, or not. Sridhar and Raghavendra [9] proved that there are at least $2^{n-4\log_2 3+1} - 1$ UPP graphs on 2^n vertices. In Section III we introduce a large class of nonisomorphic UPP graphs of order n. We prove that the size of this class is at least $2^{(2^{n+1}-3n-2)/9}$. Applications are given in Section IV.

II. Isomorphism of UPP Graphs

In this section we derive an algorithm which accepts as an input two UPP graphs $G(1)$ and $G(2)$, and determines whether $G(1)$ and $G(2)$ are isomorphic or not. In this algorithm there will be an important role for the Breadth First Search (BFS) trees from the self-loops [a vertex v is a self-loop if there exist an edge (v, v)] of the UPP graphs. Mendelsohn [10] has observed that there are exactly two self-loops in each UPP graph. For $n = 4$ the two BFS trees, from the self-loops of G_4, are illustrated in Fig. 1.

Since there are exactly two self-loops in each UPP graph, there are two possible bijections for the self-loop vertices of $G(1)$ and

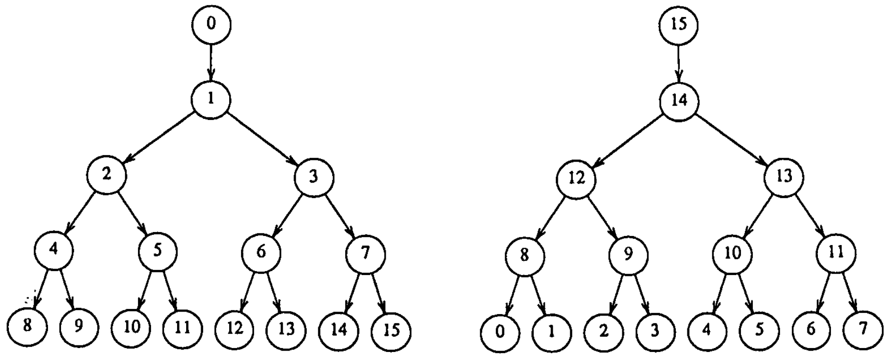

Fig. 1. The BFS trees of G_4 from the self-loop vertices.

$G(2)$. The algorithm decides if one of the bijections can achieve isomorphism between the graphs. The algorithm uses the following property of UPP graphs.

Lemma 2.1: Let v be a vertex in a UPP graph which has outgoing edges to the vertices u_1 and u_2. If w is a self-loop vertex then

$$d(u_1, w) = n \quad \text{iff } d(u_2, w) < n.$$

Proof: If $d(u_1, w) = n$ and $d(u_2, w) = n$, then there is no path of length n from v to w. If $d(u_1, w) < n$ and $d(u_2, w) < n$, then there are at least two different paths of length n from v to w. Hence, in both cases we have a contradiction to the UPP property. Thus $d(u_1, w) = n$ iff $d(u_2, w) < n$. Q.E.D.

As noted above, given two UPP graphs there are only two possible bijections for the self-loop vertices which can achieve isomorphism between the graphs. Algorithm A, given below, accepts two self-loop vertices l_1 and l_2 of $G(1)$ and $G(2)$, respectively, and tries to achieve isomorphism with the initial bijection $h(l_1) = l_2$. Then it defines the bijection for all the other vertices of the graphs by using Lemma 2.1. Namely, if h is defined for a vertex $v \in G(1)$, which has outgoing edges to u_1 and u_2, then by Lemma 2.1, u_1 and u_2 can be uniquely mapped to the vertices in $G(2)$ which have incoming edges from the vertex $h(v) \in G(2)$. In algorithm A, (A2) assigns a necessary bijection h for all the vertices of the graph which was imposed by the initial bijection for the self-loop vertices. In (A3) we check that h is a legal bijection by scanning the edge lists of $G(1)$ and $G(2)$. If the bijection is illegal then the algorithm tries the other possible initial bijection for the self-loops. If for the other possible bijection the algorithm finds that the bijection is illegal, then $G(1)$ and $G(2)$ are not isomorphic.

Algorithm A:

Let l_1, l_2 denote two self-loop vertices of $G(1)$ and $G(2)$, respectively. The initial bijection is $h(l_1) = l_2$; for each other vertex $v \in G(1)$, $h(v)$ is undefined.

(A1) Compute the distances $d(v, l_1)$ and $d(u, l_2)$ for each vertex v and u in $G(1)$ and $G(2)$, respectively, by applying the BFS procedure on the reverse graphs of $G(1)$ and $G(2)$.

(A2) Scan the vertices of $G(1)$ and $G(2)$ from l_1 and $l_2 = h(l_1)$ using BFS. Look at the stage where the BFS scans vertices v_1 and v_2 in $G(1)$, which have incoming edges from a vertex v. Let u_1 and u_2 be the vertices with incoming edges from $h(v)$ in $G(2)$.
If $d(v_1, l_1) = d(u_1, l_2)$ then $h(v_1) = u_1$; $h(v_2) = u_2$; else $h(v_1) = u_2$; $h(v_2) = u_1$.

(A3) Scan the edge lists of $G(1)$ and $G(2)$ and check for every edge $(u, v) \in G(1)$, if $(h(u), h(v)) \in G(2)$, until h is found to be illegal or all the edges were checked. If h is legal then $G(1)$ and $G(2)$ are isomorphic.

Theorem 2.2: Let $G(1)$ and $G(2)$ be two UPP graphs. By applying algorithm A with the two possible bijections for the self-loop vertices, one decides in $O(|E|)$ if $G(1)$ and $G(2)$ are isomorphic or not.

Proof: The correctness of the algorithm follows immediately from the discussion before the description of the algorithm. The complexity of each BFS in (A1) and (A2) is $O(|E|)$, and in (A3) each graph edge is checked at most once. Thus the total complexity is $O(|E|)$. Q.E.D.

III. Constructions of UPP Graphs

Sridhar and Raghavendra [9] gave a construction with a lower bound of $2^{2^{n-4}\log_2 3+1} - 1$ nonisomorphic UPP graphs. In order to describe the constructions of UPP graphs we need the following definitions and lemmas.

Let $G = (V, E)$ be a UPP graph. Two vertices u_1 and u_2 are input-compatible if there exist two vertices v_1 and v_2 such that all the vertices are distinct and $(v_1, u_1), (v_1, u_2), (v_2, u_1)$, and (v_2, u_2) are all edges in G.

Let $R^i_G(v)$ denote the set of vertices which are reachable from a vertex v with a path of length i, $1 \le i \le n$, in G.

We say that a pair of input-compatible vertices u_1 and u_2 of G satisfies property P if the following holds. Let $(u_1, x_1), (u_1, x_2), (u_2, y_1)$, and (u_2, y_2) be the outgoing edges from u_1 and u_2. Then there exist $i, j \in \{1, 2\}$ such that

$$R^{n-1}_G(x_i) = R^{n-1}_G(y_j).$$

The *exchange* operation on G (distinguished from the EXCHANGE operation) is defined as follows. Let u_1, u_2 be input-compatible vertices in G which satisfy property P, and suppose (u_1, x) and (u_2, y) are edges in E such that $R^{n-1}_G(x) = R^{n-1}_G(y)$. Then define the graph $G' = G(ex(u_1, u_2)) = (V, E')$ where

$$E' = E - \{(u_1, x), (u_2, y)\} \cup \{(u_1, y), (u_2, x)\}.$$

This is illustrated in Fig. 2.

Sridhar and Raghavendra [9] proved the following lemma.

Lemma 3.1: If the graph G' was obtained from a UPP graph G by the exchange operation, then G' is a UPP graph.

Sridhar and Raghavendra [9] considered a set S with 2^{n-3} pairs of input-compatible vertices in G_n which consists of the pairs with the binary form $\{b_1 \cdots b_{n-3}010, b_1 \cdots b_{n-3}011\}$. They proved that by applying the exchange operation on all the input-compatible vertices of any subset of S, such that $\{b_1 \cdots b_{n-3}010, b_1 \cdots b_{n-3}011\} \in$

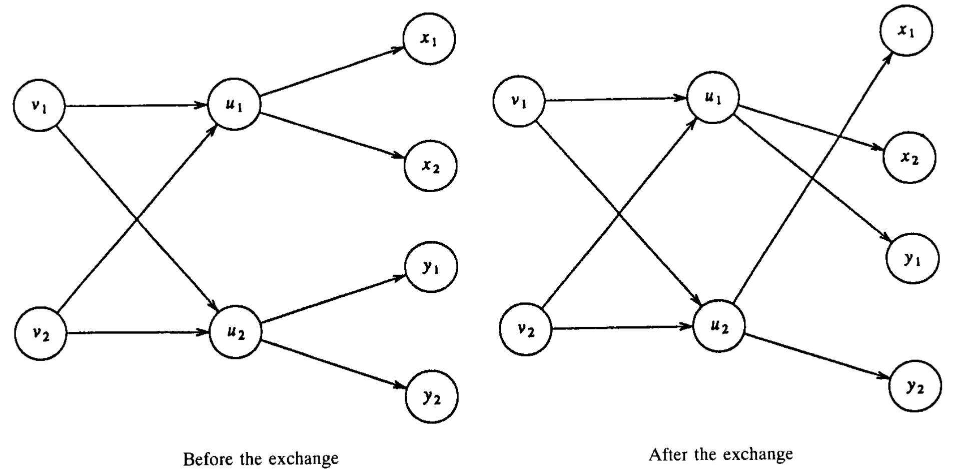

Fig. 2. The Exchange operation.

S iff $\{\bar{b}_1 \cdots b_{n-3}010, \bar{b}_1 \cdots b_{n-3}011\} \notin S$, one derives nonisomorphic UPP graphs. In addition, all the reverse graphs are not isomorphic among themselves and with the original UPP graphs. The only exception is G_n, which is isomorphic to its reverse. Thus, obtaining at least $2^{2^{n-4}\log_2 3+1} - 1$ nonisomorphic UPP graphs [9].

In this section we show that the class of UPP graphs is considerably richer. In the sequel we will refer only to input-compatible vertices which satisfy property P.

Two pairs of input-compatible vertices $S_{uv} = \{u, v\}$ and $S_{xy} = \{x, y\}$ with outgoing edges to the sets $S_{ouv} = \{u_1, u_2, v_1, v_2\}$ and $S_{oxy} = \{x_1, x_2, y_1, y_2\}$, respectively, are called independent if

$$S_{uv} \cap S_{oxy} = \emptyset \quad \text{and} \quad S_{xy} \cap S_{ouv} = \emptyset.$$

A set S of input-compatible vertex pairs is called independent if every two pairs of S are independent.

Lemma 3.2 [9]: Let G be a UPP graph and G' be the graph derived from G by an *exchange* operation with input-compatible pair of vertices v and w. Then for each $i \geq 2$, for every vertex $x \notin \{v, w\}$, and every vertex y,

$$y \in R^i_{G'}(x) \quad \text{iff } y \in R^i_G(x).$$

Lemma 3.3: Let G be a UPP graph and let S be an independent set of input-compatible vertex pairs. If $\{u, v\} \in S$, then in $G' = G(ex(u, v))$ the set $S' = S - \{\{u, v\}\}$ is an independent set of input-compatible vertex pairs.

Proof: Since the edges (u, u_1) and (v, v_1) were removed, all the pairs of vertices which were input-compatible except for pairs from $S_{ouv} = \{u_1, u_2, v_1, v_2\}$ remain input-compatible. By Lemma 3.2 each pair $\{x, y\} \in S'$ satisfies property P. Q.E.D.

We will denote by $G(ex(S))$ the UPP graph obtained from G by performing all the exchanges defined by the set S in arbitrary order. Note that the obtained graph is oblivious to the order in which the exchanges are executed.

Let S be a set of input-compatible vertex pairs. The *complement* $\overline{S}$ of S is defined by

$$\overline{S} = \{\{u, v\} \mid \{N - 1 - v, N - 1 - u\} \in S\}.$$

The following lemma can be easily verified.

Lemma 3.4: Let S be an independent set of input-compatible vertex pairs. Then $G_n(ex(S))$ is isomorphic to $G_n(ex(\overline{S}))$ under the bijection

$$\forall\, v \in G_n(ex(\overline{S})), h(v) = N - 1 - v.$$

Throughout this paper we will consider only exchanges in which the edges (x, y) which are removed, have the property that y is a vertex labeled with an even integer. This is not a limitation on the number of nonisomorphic graphs, and it can be understood from Fig. 3. By making the substitution $f(2i) = (2i + 1)$ and $f(2i + 1) = (2i)$ we get the effect of exchanges in which in the edges (x, y) which are removed, y is labeled by an odd integer.

The following lemmas will enable us to prove that given two different independent sets of input-compatible vertex pairs, S_1 and S_2 such that $S_1 \neq \overline{S_2}$, the graphs $G(1) = G_n(ex(S_1))$ and $G(2) = G_n(ex(S_2))$ are nonisomorphic.

Proposition 3.5: In G_n, $d(2i, 0) < n$, $d(2i, N - 1) = n$, $d(2i + 1, 0) = n$, and $d(2i + 1, N - 1) < n$.

Proof: Follows immediately from the observation that the binary representation of $2i$ ends with a ZERO and that $2i + 1$ ends with a ONE. Q.E.D.

Lemma 3.6: Let S be an independent set of input-compatible vertex pairs. In $G_n(ex(S))$, for each $\{2i, 2i + 1\} \in S$ we have that $d(2i, 0) = n$, $d(2i + 1, 0) < n$, and all other distances to 0 and $N - 1$ remain as in G_n.

Proof: Let $\bigcirc\!\!\overset{j}{(l_1,l_2)}$ denote vertex j with $d(j, 0) = l_1$, and $d(j, N - 1) = l_2$. By Proposition 3.5 we can deduce that for each pair of vertices $\{2i, 2i + 1\}$ the labeling is as illustrated in Fig. 3(a). By Lemma 3.2 all the distances from the vertices beside vertices $2i$ and $2i + 1$ remain the same as in G_n, when the exchange operation is applied on $2i$ and $2i + 1$. Considering this labeling and the exchange which was performed, one can easily verify that the new labeling is as in Fig 3(b). Performing all the exchanges and reapplying Lemma 3.2 obtains the required result. Q.E.D.

Note that after the exchange operation, $d(2i, 0) = d(2i, N - 1) = n$, and $d(2i + 1, 0) < n$, $d(2i + 1, N - 1) < n$. This property will be used in the proof of nonisomorphism.

Assume that the two self-loops of two isomorphic UPP graphs, $G(1)$ and $G(2)$, are 0 and $N - 1$. As mentioned in Section II, there are two possible bijections between $G(1)$ and $G(2)$. Let h_0 be a bijection in which $h_0(0) = 0$ and $h_0(N - 1) = N - 1$, and h_1 is the bijection in which $h_1(0) = N - 1$, $h_1(N - 1) = 0$.

Lemma 3.7: Let S_1 and S_2 be two different sets of input-compatible vertex pairs. Then there is no legal bijection under h_0 for

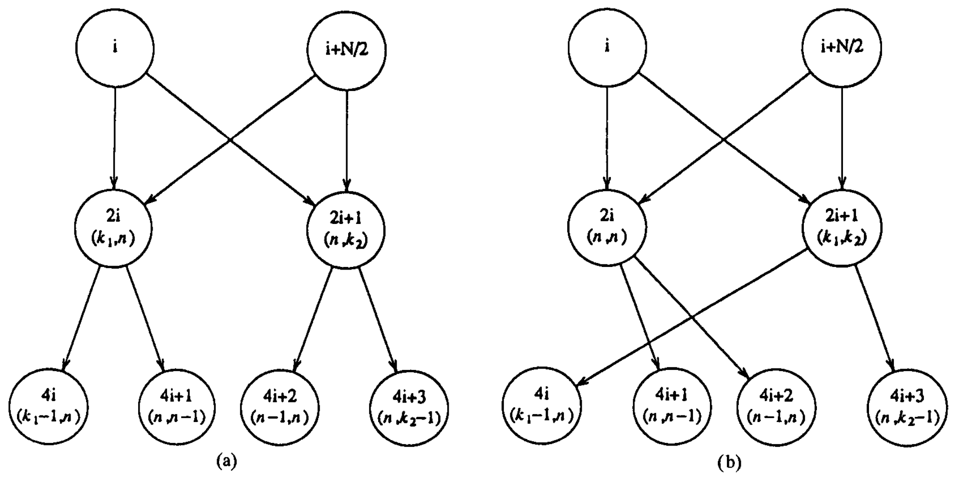

Fig. 3. Labeling before and after the exchange operation. (a) Labeling before the exchange. (b) Labeling after the exchange.

which $G(1) = G_n(ex(S_1))$ is isomorphic to $G(2) = G_n(ex(S_2))$.

Proof: Assume Algorithm A finds a legal bijection h between $G(1)$ and $G(2)$ under h_0. If h is the identity mapping, namely for every vertex v, $h(v) = v$, then let $v = 2i$, $u = 2i + 1$ be an input-compatible vertex pair $\{v, u\} \in S_1$, $\{v, u\} \notin S_2$. The sons of v in $G(1)$ are $4u + 2$ and $4u + 1$, while in $G(2)$ the sons are $4u$ and $4u + 1$, hence h is not a legal bijection.

Assume h is not the identity mapping. Let v be the first vertex in $G(1)$ that is assigned by h such that $h(v) \neq v$. We claim and prove that this mapping is due to an exchange operation which is only in one of the graphs. Denote the father of v in the BFS tree of $G(1)$ by $f(v)$; $h(v)$ is defined by traversing the edge $f(v) \rightarrow v$ in the BFS tree of $G(1)$ and its corresponding edge in $G(2)$. By our assumption $h(f(v)) = f(v)$. If $f(v)$ and its companion (namely $f(v) + 1$ if $f(v)$ is even, and $f(v) - 1$ if $f(v)$ is odd) are not in S_1 and not in S_2, then both sons of $f(v)$ in $G(1)$ and $G(2)$ are the same vertices, and by Proposition 3.5 and Lemma 3.6 their distances to the vertices 0 and $N - 1$ are as in G_n. If $f(v)$ and its companion belong to S_1 and S_2 then again in both graphs the sons of $f(v)$ are the same vertices, and their distances to the vertices 0 and $N - 1$ are changed as described in Lemma 3.6. In both cases, algorithm A would result with a mapping of $h(v) = v$, and thus we conclude that $f(v)$ and its companion belong only to one of the independent vertex pair sets due to the mapping $h(v) \neq v$. Without loss of generality assume the pair belongs to S_1. Then by Lemma 3.6, in $G(1)$, $d_{G(1)}(f(v), 0) = d_{G(1)}(f(v), N - 1) = n$ or $d_{G(1)}(f(v), 0) < n$ and $d_{G(1)}(f(v), N - 1) < n$. While in $G(2)$, $d_{G(2)}(f(v), 0) = n$ and $d_{G(2)}(f(v), N - 1) < n$ or $d_{G(2)}(f(v), 0) < n$ and $d_{G(2)}(f(v), N - 1) = n$. Hence, h is not a legal bijection in contradiction to our assumption, which concludes that under h_0, $G_n(ex(S_1))$ and $G_n(ex(S_2))$ are not isomorphic.

Q.E.D.

Lemma 3.8: Let S_1 and S_2 be two independent sets of input-compatible vertex pairs. If $S_1 \neq S_2$ and $S_1 \neq \overline{S_2}$, then $G_n(ex(S_1))$ is not isomorphic to $G_n(ex(S_2))$.

Proof: a) By Lemma 3.7 there is no legal bijection between $G_n(ex(S_1))$ and $G_n(ex(S_2))$ under h_0.

b) By Lemma 3.4, $G_n(ex(S_2))$ is isomorphic to $G_n\left(ex\left(\overline{S_2}\right)\right)$ under h_1.

c) By Lemma 3.7, $G_n(ex(S_1))$ in not isomorphic to $G_n\left(ex\left(\overline{S_2}\right)\right)$ under h_0. Hence, by b) and c), $G_n(ex(S_1))$ is not isomorphic to $G_n(ex(S_2))$ under h_1. Q.E.D.

Now, given two independent sets of input-compatible vertex pairs S_1 and S_2, it is clear that $G_n(ex(S_1))^R$ is not isomorphic to $G_n(ex(S_2))^R$, unless $S_1 = S_2$ or $S_1 = \overline{S_2}$. We also want to show that the intersection between the set of UPP graphs, obtained from all the independent sets and the set of their reverses, consists only of one UPP graph, G_n.

Lemma 3.9: Let S_1 and S_2 be two sets of independent input-compatible vertex pairs. Then $G' = G_n(ex(S_1))$ is not isomorphic to $G_n(ex(S_2))^R$, unless $S_1 = S_2 = \emptyset$.

Proof: Assuming that $S_2 \neq \emptyset$ then by Proposition 3.5 and Lemma 3.6, there exists a vertex u in $G_n(ex(S_2))$ such that $d(u, 0) = d(u, N - 1) = n$. Therefore in $G_n(ex(S_2))^R$, $d(0, u) = d(N - 1, u) = n$. Similarly to Proposition 3.5 we have that for each vertex v in G_n, $d(0, v) = n$ iff $d(N - 1, v) < n$. By Lemma 3.2 we have that $R^i_{G'}(0) = R^i_{G_n}(0)$ and $R^i_{G'}(N - 1) = R^i_{G_n}(N - 1)$, for $2 \leq i \leq n$. Also, it is clear by the definition of the exchange operation that $R^1_{G'}(0) = R^1_{G_n}(0)$ and $R^1_{G'}(N - 1) = R^1_{G_n}(N - 1)$. Hence, in G' for each v, $d(0, v) = n$ iff $d(N - 1, v) < n$. Since in $G_n(ex(S_2))^R$, $d(0, u) = d(N - 1, u) = n$, $G_n(ex(S_1))$ is not isomorphic to $G_n(ex(S_2))^R$, unless $S_1 = S_2 = \emptyset$. Q.E.D.

From Lemma 3.8 and Lemma 3.9 we infer a concluding theorem.

Theorem 3.10: Let S_1 and S_2 be two independent sets of input-compatible vertex pairs. $G_n(ex(S_1))$ is not isomorphic to $G_n(ex(S_2))$ unless $S_1 = S_2$ or $S_1 = \overline{S_2}$.. $G_n(ex(S_1))$ is not isomorphic to $G_n(ex(S_2))^R$ unless $S_1 = S_2 = \emptyset$.

Using Theorem 3.10 we want to derive a lower bound on the number of nonisomorphic UPP graphs. Let GIS (Graphs from Independent Sets) denote the set of nonisomorphic UPP graphs obtained by Theorem 3.10. We will construct a dependency graph, in which each vertex represents an input-compatible vertex pair (u, v) in G_n. Between vertices (u_1, v_1) and (u_2, v_2) there is an edge if (u_1, v_1) and (u_2, v_2) are dependent, i.e., each vertex $(i, i+1)$ where i is even, $i \neq 0$, $i \neq N - 2$, is connected by edges to the vertices $(2i, 2i + 1)$ and $(2i + 2, 2i + 3)$. One can easily verify, by using the substitution $f(i, i + 1) = i/2$, i even, $i \neq 0$, and $i \neq N - 2$, that this dependency graph is the underlying graph of G_{n-1}, without the vertices 0 and $(N/2) - 1$. By Theorem 3.10 each two independent sets of vertices in this graph correspond to two nonisomorphic UPP graphs, unless their corresponding sets of input-compatible vertices are complements. Those nonisomorphic graphs and their reverses (except for $(G_n)^R$) induce a set of nonisomorphic UPP graphs. Bryant and Fredricksen [13] proved that the size of the maximum independent set in G_n is at

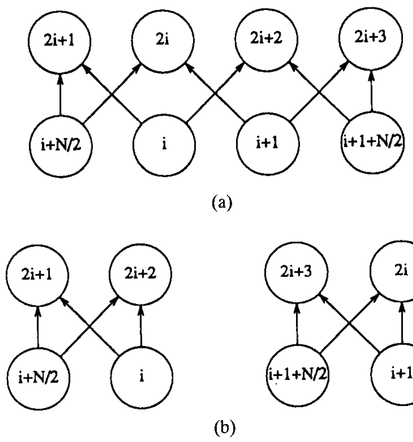

Fig. 4. The possible alternating cycles in GIS. (a) After one exchange. (b) After the second exchange.

least $(2^{n+2} - 3n - 5)/9$. Therefore we infer the following theorem.

Theorem 3.11: The number of UPP graphs induced by Theorem 3.10 is at least $2^{(2^{n+1}-3n-2)/9}$.

Although this number is significantly larger than the $2^{2^{n-4}\log_2 3+1} - 1$ UPP graphs obtained by Sridhar and Raghavendra [9], we know that this number is considerably smaller than the actual number of nonisomorphic UPP graphs in GIS. Moreover, we claim that the class of UPP graphs is even richer than graphs in GIS. In [14] other constructions for UPP graphs are given and we know that other nonisomorphic UPP graphs exist. But, we were not able to improve the lower bound given by Theorem 3.11. A simple upper bound on the number of UPP graphs is derived from the facts that each vertex lies on a directed cycle whose length is a divisor of n and Mendelsohn theorem [10] that the number of cycles of length $k \leq n$ is equal in all the UPP graphs on 2^n vertices. Therefore, each UPP graph has a set of disjoint cycles of lengths which divide n and each two sets in two nonisomorphic graphs are isomorphic. Hence, only 2^n edges have to be arranged and we have

Theorem 3.12: The number of UPP graphs on 2^n vertices is less than $(2^n)!$.

Finally, an important property of the graphs in GIS is given in the following lemma.

Lemma 3.13: All the alternating cycles in the UPP graphs in GIS have length 4 or 8.

Proof: By some exchange operations on G_n, which include the input-compatible vertices $\{i, i+1\}$ and does not include any pair $\{i+N/2, i+N/2+1\}$, we obtain the alternating cycle as in Fig. 4(a). It is easy to verify that only an exchange operation with $\{i+N/2, i+N/2+1\}$ can change this alternating cycle by splitting it into the two alternating cycles of length 4 as in Fig. 4(b). Q.E.D.

IV. Applications

Some of the UPP networks derived from the UPP graphs obtained in Section III were found to be useful in realization of arbitrary permutations [14]. In this section we derive the connection between the realization of permutations of UPP networks and on UMFA networks. From UPP graphs we generate UMFA networks by the following two constructions.

Construction 4.1: Given a UPP graph G with 2^n vertices one can construct an MI network with l stages such that each stage has 2^n SE's. There is a directed edge from SE i in stage k to SE j in stage $k+1$ if there is a directed edge from vertex i to vertex j in G.

Construction 4.2: Given a UPP graph G with 2^{n+1} vertices such that all the alternating cycles have length 4 (buddy property), we can construct an MI network with l stages such that each stage has 2^n SE's labeled by (a, b, c, d), where in G both a and b have outgoing edges to c and d. There is a directed edge from SE (a, b, c, d) in stage k to SE (e, f, g, h) in stage $k+1$ if either $c \in \{e, f\}$ or $d \in \{e, f\}$.

Clearly, both constructions produce UMFA networks. Moreover, by applying Construction 4.1 on a UPP graph G we obtain the same MI network produced by applying Construction 4.2 on the line graph [15] of G. Therefore, if one can find an algorithm that realizes an arbitrary permutation on a UPP network NE, we can use this algorithm for rearrangeability of the corresponding UMFA network.

References

[1] D.H. Lawrie, "Access and alignment of data in an array processor," *IEEE Trans. Comput.,* vol. C-24, pp. 1145–1155, Dec. 1975.
[2] K.E. Batcher, "The flip network in STARAN," in *Proc. Int. Conf. Parallel Processing,* Aug. 1976, pp. 65–71.
[3] C.L. Wu and T.Y. Feng, "On a class of multistage interconnection networks," *IEEE Trans. Comput.* vol. C-29, pp. 694–702, Aug. 1980.
[4] M.C. Pease, "The indirect binary cube microprocessors array," *IEEE Trans. Comput.,* vol. C-26, pp. 458–473, May 1977.
[5] T. Feng, "Data manipulating functions in parallel processors and their implementations," *IEEE Trans. Comput.,* vol. C-23, pp. 309–318, Mar. 1974.
[6] C.P. Kruskal and M. Snir, "A unified theory of interconnection network structure," *Theoret. Comput. Sci.,* vol. 48, pp. 75–94, 1986.
[7] J.C. Bermond, J.M. Fourneau, and A. Jean-Marie, "Equivalence of multistage interconnection networks," *Inform. Processing Lett.,* vol. 26, pp. 45–50, Sept. 1987.
[8] C.R. Goke and G.J. Lipovski, "Banyan networks for partitioning multiprocessor systems," in *Proc. 1st Annu. Symp. Comput. Architecture,* Dec. 1973, pp. 21–28.
[9] M.A. Sridhar and C.S. Raghavendra, "Uniform minimal full-access networks," *J. Parallel Distributed Comput.,* vol. 5, pp. 383–403, 1988.
[10] N.S. Mendelsohn, "Directed graphs with unique path property," in *Combinatorial Theory and its Applications,* P. Erdos, A. Renyi, and V.T. Sos, Eds. Amsterdam: The Netherlands, North-Holland, 1970, pp. 783–799.
[11] N.G. de Bruijn, "A combinatorial problem," *Nederl. Akad. Wetensch. Proc.,* vol. 49, pp. 758–764, 1946.
[12] D.P. Agrawal, "Graph theoretical analysis and design of multistage interconnection networks," *IEEE Trans. Comput.,* vol. C-32, pp. 637–648, July 1983.
[13] R.D. Bryant and H. Fredricksen, "Covering the de Bruijn graph," *Discrete Math.,* vol. 89, pp. 133–148, 1991.
[14] D. Goldfeld, "Equivalence of interconnection networks," M.Sc. thesis, Technion, May 1989.
[15] R.L. Hemminger and L.W. Beineke, "Line graphs and line digraphs," in *Selected Topics in Graph Theory,* L.W. Beineke and R.J. Wilson, Eds. London, England: Academic, 1978.