Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

J. Parallel Distrib. Comput. 64 (2004) 135–150

Journal of Parallel and Distributed Computing

http://www.elsevier.com/locate/jpdc

Efficient algorithms for checking the equivalence of multistage interconnection networks

Tiziana Calamoneri* and Annalisa Massini

Dip. di Informatica, Università di Roma "La Sapienza", via Salaria 113-00198 Rome, Italy

Abstract

In this paper we study the topological equivalence problem of multistage interconnection networks (MINs). We prove a new characterization of topologically equivalent MINs by means of a novel approach. Applying this characterization to $\log N$ stage MINs we completely describe the equivalence class which the Reverse Baseline belongs to. Most important, we apply the characterization to $(2\log N - 1)$ stage MINs obtained as concatenation of two $\log N$ stage Reverse Baseline equivalent MINs: in this way, we deduce an $O(N \log N)$ time algorithm testing the equivalence of two such MINs. This result substantially improves the time complexity of the previously known algorithms ($O(N^4 \log N)$). Finally, we determine the number of different equivalence classes of $(2\log N - 1)$ stage MINs and we characterize each of them.

Keywords: Multistage interconnection networks; Topological equivalence; Layered cross product; Layered graphs

1. Introduction

Experience with the design and use of parallel computers indicates that the efficiency of a parallel computer (among other things) is largely dependent on the properties of the interconnection network, i.e. the device devoted to the information exchange between processors and memories. Namely, the interconnection network not only affects the hardware architecture but also the nature of the system software (such as the network operating system). Several topologies have been proposed to realize interconnection networks; in particular, multistage interconnection networks (MINs) have been widely studied by researchers (for a survey, see [8,19]) and implemented in some practical systems for the efficient communication schemes they provide. Usually MINs with N inputs and N outputs, $N = 2^n$, and stages with $\frac{N}{2}$ switching elements of size 2×2 are considered.

MINs consisting of $\log N$ stages such as Omega [17], Flip [2], Indirect Binary Cube [21], Modified Data Manipulator [11], Baseline and Reverse Baseline [22], Butterfly and Reverse Butterfly [19] networks, have the same underlying graph and present attractive advantages: efficient routing algorithms, partitionability, small number of switching elements. Unfortunately, these networks are *blocking*.

For this reason, $(2\log N - 1)$ stage MINs have been intensively studied. They are obtained by concatenating two $\log N$ stage MINs with the center stage overlapped; Beneš network [3] is an example of this class of MINs and it is usually represented as the concatenation of a Baseline and a Reverse Baseline.

In the past two decades the binary relation of *topological equivalence* between two different MINs has been widely investigated because, although many MINs appear completely different (in the sense that their usual graphical representations are not the same and it is not trivial to find a 1-1 function identifying the networks as N grows), they are intrinsically the same, i.e. the underlying graph is the same; hence, the understanding of this equivalence relation makes it possible to apply the routing scheme designed for a MIN to an equivalent one and to develop more general routing algorithms useful for all MINs in the same equivalence class, independently from their aspect.

It is well known that the general problem of deciding whether two graphs are isomorphic is easily in NP, but it is not known to be in P and it is not known to be NP-complete [13,15]. Nevertheless, the question restricted to

*Corresponding author.
E-mail addresses: calamo@dsi.uniroma1.it (T. Calamoneri), massini@dsi.uniroma1.it (A. Massini).

0743-7315/$ - see front matter © 2003 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2003.09.003

particular graphs is easier. Namely, let us look at MINs: for what concerns $\log N$ stage MINs, Wu and Feng [22] present the first formal definition of topological equivalence and prove that the six most common $\log N$ stage MINs (those mentioned above) are topologically equivalent: they exhibit isomorphisms between all pairs of networks, but do not give any characterization of the equivalence class. Another approach is considered by Agrawal [1], but unfortunately it is correct only for $\log N$ stage MINs of small dimension. A revised version is proposed by Bermond et al. [4]: they give more general properties to check the topological equivalence. Hu et al. [14] present a simplified checking equivalence algorithm based on a marking scheme of nodes whose time complexity is $O(N \log N)$, that is optimal.

A lot of efforts have been expended also in studying topological equivalence of $(2 \log N - 1)$ stage MINs. Wu and Feng [23] extend their equivalence properties for $\log N$ stage MINs to prove that two-passes of a Reverse Baseline network has the same routing capability and is equivalent to the Beneš network. Lee [18] proves that an Omega network concatenated with its reverse is equivalent to the Beneš network. Yeh and Feng [24] propose a coding scheme to check whether a given $(2 \log N - 1)$ stage MIN is topologically equivalent to the Beneš network, but this scheme leaves many cases uncovered. Feng et al. [12] study 36 common topologies obtained as concatenation of two $\log N$ stage MINs and classify them into two equivalence classes. Hu et al. [14] investigate whether a $(2 \log N - 1)$ stage MIN is the concatenation of two $\log N$ stage MINs and give an algorithm to determine whether two given $(2 \log N - 1)$ stage MINs are topologically equivalent in $O(N^4 \log N)$ time.

In almost all the cited papers, the studied MINs are considered as strictly dependent from their more usual graphical representation. In this work we approach the topological equivalence problem from a novel point of view: by means of the layered cross product [10], we do not deal with the appearance of the considered MINs but we consider the underlying graph to extract the structural properties useful for the equivalence. This different approach allows us to propose a new characterization for the topological equivalence of MINs that is independent from the particular appearance of each MIN. This very general characterization achieves interesting results when applied to $(2 \log N - 1)$ stage MINs.

Namely, we deal with MINs obtained as concatenation of two Reverse Baseline equivalent MINs; first, we provide an optimal algorithm (i.e. running in $O(N \log N)$ time) testing the equivalence of two such MINs. This algorithm improves of a factor $O(N^3)$ the time complexity of the previously known one [14]. Then, we determine the number of different equivalence classes and we characterize each of them. These results definitely close the topological equivalence problem on this kind of $(2 \log N - 1)$ stage MINs.

The rest of this paper is organized as follows. In Section 2 we give some preliminary definitions. Section 3 is devoted to the new characterization of topological equivalence of MINs. In Section 4 we specialize this characterization to $\log N$ stage MINs and we provide an $O(N \log N)$ time algorithm, checking the equivalence of a MIN with the Reverse Baseline. Although this result does not improve the time complexity of the previously known one [14], its interest lies in the originality of the approach, and we detail it since it is preliminary to the algorithm concerning $(2 \log N - 1)$ stage MINs. In Section 5 we give an $O(N \log N)$ time algorithm to check the topological equivalence of two MINs obtained as concatenation of two Baseline equivalent MINs and we characterize the equivalence classes. Finally, Section 6 is devoted to some final considerations and open problems.

2. Preliminary definitions

In this section we give some basic definitions and preliminary results, useful for the comprehension of the rest of this paper.

Definition 2.1. An *l-layered graph*, $G = (V_1, V_2, \ldots, V_l, E)$ consists of l layers of nodes; V_i is the (non-empty) set of nodes in layer i, where $1 \leqslant i \leqslant l$; E is a set of edges: every edge connects nodes of two adjacent layers.

Observe that a rooted tree T of height h is a particular case of h-layered graph, where layer i is defined either as the set of nodes having distance $i-1$ from the root or as the set of nodes having distance $h-i$ from the root. From now on, we shall call a complete binary tree T by means of Δ or ∇ according to whether the first or the second way of defining layers is chosen (the notation comes from the usual graphic representation of such trees—see Fig. 1a and b).

The $N \times N$ *multistage interconnection network (N-MIN)* is another particular l-layered graph, where layers coincide with stages, $|V_i| = \frac{N}{2}$ and any node in V_i, representing a switching element of size 2×2, is adjacent to exactly two nodes in V_{i-1} and V_{i+1}, $2 \leqslant i \leqslant l-1$, and the edge set is partitioned into $l-1$ subsets, called *edge-stages*, $E_1, E_2, \ldots, E_{l-1}$ such that E_i contains N edges, that is all edges between V_{i-1} and V_i. Each node of the first stage is also connected to a pair of *inputs* and each node of the last stage is also connected to a pair of *outputs*. Then an N-MIN contains N inputs and N outputs.

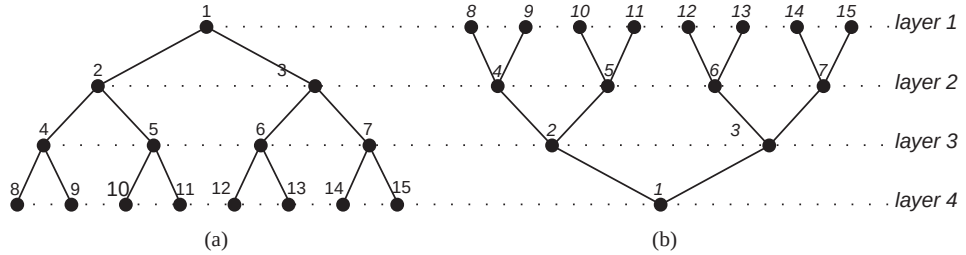

Fig. 1. A Δ and a ∇, both of height $h = 4$.

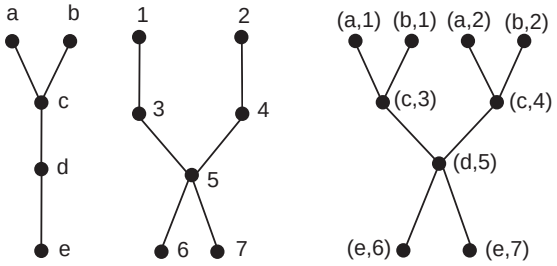

Fig. 2. The LCP of two graphs.

Definition 2.2 (Even and Litman [10]). Let $G' = (V'_1, V'_2, \ldots, V'_l, E')$ and $G'' = (V''_1, V''_2, \ldots, V''_l, E'')$ be two l-layered graphs. Their *layered cross product* (LCP for short), $G' \otimes G''$ is a l-layered graph $G = (V_1, V_2, \ldots, V_l, E)$ where a node is a pair, since V_i is the cartesian product of V'_i and V''_i, $1 \leqslant i \leqslant l$, and an edge $\langle (u', u''), (v', v'') \rangle$ belongs to E if and only if $\langle u', v' \rangle \in E'$ and $\langle u'', v'' \rangle \in E''$. G' and G'' are called the *first* and the *second factor* of G, respectively. An example is depicted in Fig. 2.

We will call *decomposition in factors* the inverse operation of the LCP.

Even and Litman [10] introduce the LCP technique to show that it enables the representation of several well-known MINs as a layered cross product of simple networks. They also show that this decomposed representation considerably simplifies some proofs of properties of such MINs and may be useful in the analysis and in the synthesis of the decomposed MINs. Successively, LCP has been used to investigate on some problems such as grid embedding [9] and interval routing [5,16]. In this paper we extend the use of LCP to investigate on the topological equivalence problem of MINs.

Fact 1. *Simple path is the neutral element of LCP operation.*

Lemma 2. *Given two l-layered graphs, having c_1 and c_2 connected components, respectively, their LCP has $c_1 \cdot c_2$ connected components.*

Proof. It immediately follows from the definition of LCP. □

Given any two layers i and $j, i<j$, we say that a l-layered graph G *has a simple cycle from i to j* if the shortest cycle passing through any node at layer i and any node at layer j is $2(j-i)$ long; in other words, G has a simple cycle from i to j if there exists a cycle passing through all layers from i to j and does not pass through layers $i-1$ and $j+1$; furthermore, it does not exist a layer $j'(i'), i<j'<j (i<i'<j)$ such that G has a simple cycle from i to j' (from i' to j).

Lemma 3. *Given two l-layered graphs G_1 and G_2, if G_1 (G_2) has a simple cycle from i to j for some i and $j, i<j$, and G_2 (G_1) has a simple cycle from i to $j' \geqslant j$, then also the LCP of G_1 and G_2 has a simple cycle from i to j.*

Proof. It is easy to see that if G_1 (G_2) has a simple cycle from i to j and $G_2(G_1)$ has a longer cycle in correspondence of the same interval of layers, then also $G_1 \otimes G_2$ has a cycle passing through all layers from i to j and $2(j-i)$ long. From the definition of LCP it follows that it cannot exist in $G_1 \otimes G_2$ a shortest cycle from i to a layer $j'' < j$. □

Lemma 4 (Even and Litman [10]). *The LCP of a Δ and a ∇, both of them with $\frac{N}{2}$ leaves, outputs an N input Butterfly network.*

Paz [20] provides a theory of decomposition into prime factors (i.e. not further decomposable graphs) of MINs, based on matrix notation. We now recall the following three graphs (see Fig. 3) belonging to the set of prime graphs described in [20]:

- the l-layered graph Λ_i, $1 \leqslant i \leqslant l-1$, has a simple path from layer 1 to layer i (null if $i = 1$), a fork between layers i and $i+1$ and two parallel simple paths from layer $i+1$ to layer l (null if $i = l-1$);
- the l-layered graph V_i, $1 \leqslant i \leqslant l-1$, has two parallel simple paths from layer 1 to layer i (null if $i = 1$), a junction between layers i and $i+1$ and a simple path from layer $i+1$ to layer l (null if $i = l-1$);
- the l-layered graph Φ_{ij}, $1 \leqslant i < j \leqslant l-1$, has a simple path from layer 1 to layer i (null if $i = 1$), a fork between layers i and $i+1$, two parallel simple paths from layer $i+1$ to layer j (null if $j = i+1$), a junction

between layers j and $j+1$ and a simple path from layer $j+1$ to layer l (null if $j = l-1$).

Prime graphs Λ_i and V_i can be used to decompose in factors binary trees; namely, a Δ of height h can be decomposed as $\Lambda_1 \otimes \Lambda_2 \otimes \cdots \Lambda_{h-1}$ and a ∇ of height h can be decomposed as $V_1 \otimes V_2 \otimes \cdots V_{h-1}$.

Now, consider graph $X_{i,j}, 1 \leqslant i \leqslant l-1, l \leqslant j \leqslant 2l-2$, defined as the $(2l-1)$-layered graph having a pair of parallel simple paths from layer 1 to layer $2l-1$, and two pairs of edges crossing between layers i and $i+1$ and between layers j and $j+1$ (see Fig. 4).

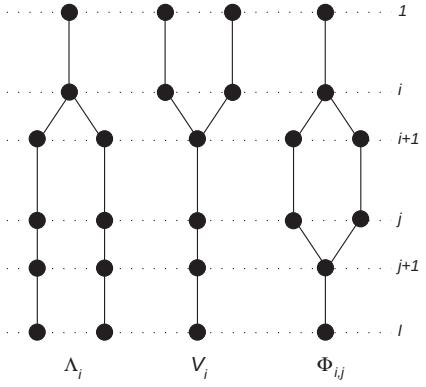

Fig. 3. Three prime graphs.

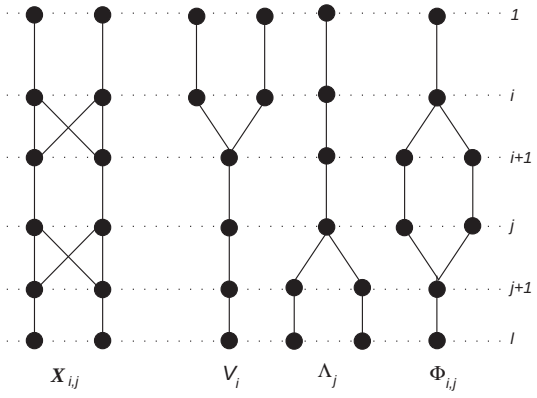

Fig. 4. $X_{i,j}$ as LCP of V_i, Λ_j and Φ_{ij}.

It is straightforward to see that $X_{i,j} = V_i \otimes \Lambda_j \otimes \Phi_{ij}$ (see Fig. 4).

The aim of this paper is to exploit the concept of LCP to study the topological equivalence problem of MINs. Hence, we recall here the definition.

Definition 2.3. Any two s stage N-MINs G' and G'' are *topologically equivalent* (or simply *equivalent*) if and only if there is an isomorphic mapping ψ such that:

- for any stage i $(i = 1, \ldots, s)$ if $v \in V_i(G')$ then $\psi(v) \in V_i(G'')$,
- for any edge-stage j $(j = 1, \ldots, s-1)$ if $\langle u, v \rangle \in E_j(G')$ then $\langle \psi(u), \psi(v) \rangle \in E_j(G'')$.

Observation 1. *The concept of topological equivalence is different from functional equivalence. In fact two N-MINs are functionally equivalent if they have the capability of always performing the same set of assignments [7]. Hence, all rearrangeable N-MINs are functionally equivalent though not necessarily topologically equivalent; on the contrary, not rearrangeable N-MINs could be topologically equivalent but not functionally equivalent.*

Observation 2. *Two N-MINs topologically equivalent are isomorphic.*

The LCP is commutative, therefore from now on, when we speak about topological equivalence between the first and the second factors of two N-MINs $G' = G'_1 \otimes G'_2$ and $G'' = G''_1 \otimes G''_2$ we mean that G'_1 and G'_2 are equivalent either to G''_1 and G''_2, or to G''_2 and G''_1, respectively.

In the rest of the paper we point out our attention on N-MINs that are decomposable as LCP of two graphs; therefore, when we speak about LCP, we implicitly assume that the result is an N-MIN. Observe that the inputs and outputs of N-MINs are not involved in the LCP, but it is not restrictive to add them at the end of the computation of the LCP (see Fig. 5).

We conclude these preliminaries by recalling some definitions that will be useful in Section 3.

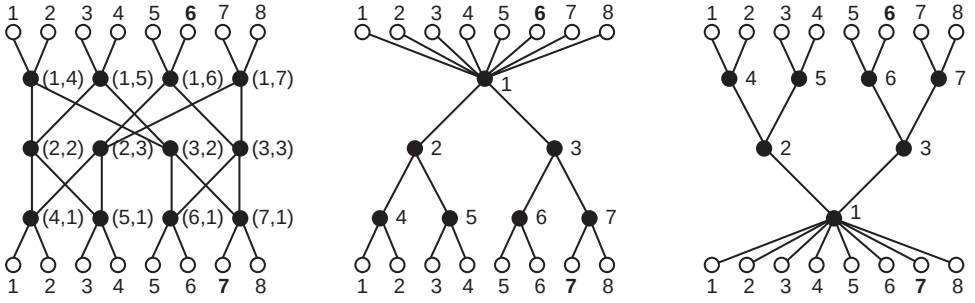

Fig. 5. A baseline network obtained as LCP of a Δ and a ∇, with inputs and outputs.

Definition 2.4. An N-MIN has the *Banyan property* if and only if for any input and any output there exists a unique path connecting them, passing through each stage once.

Lemma 5 (Even and Litman [10]). *The LCP operation yields a Banyan graph if and only if each of its factors is Banyan.*

3. A new characterization for the equivalence of N-MINs based on LCP

In this section we propose a new general characterization of N-MINs' topological equivalence. Then, we will exploit this result to design efficient algorithms for studying the topological equivalence of N-MINs.

Although the following characterization theorem can be stated for general layered graphs, we restrict it to N-MINs.

Theorem 6. *Let G' and G'' be two s stage N-MINs, and let G' decomposable as $G'_1 \otimes G'_2$. Then G'' is topologically equivalent to G' if and only if G'' can be decomposed as $G'_1 \otimes G'_2$.*

Proof. ($\Leftarrow$) If G'' can be decomposed as $G'_1 \otimes G'_2$, then G' and G'' are equivalent in view of the definition of LCP (see Definition 2.2).

($\Rightarrow$) If G' and G'' are topologically equivalent, then they are isomorphic (see Observation 2) and hence they can be decomposed into the same factors. □

Corollary 7. *Given two N-MINs $G' = G'_1 \otimes G'_2$ and $G'' = G''_1 \otimes G''_2$, they are topologically equivalent if their factors are topologically equivalent.*

Since, in general, the decomposition in factors is not unique, we have to conveniently decompose the N-MINs to check their equivalence by checking the equivalence of their factors. This decomposition operation is helpful since the factors of an N-MIN G are simpler graphs than G itself, and therefore checking the equivalence between factors may be much easier than checking the equivalence between N-MINs.

The next two sections illustrate how this very general characterization can be applied to $\log N$ and $(2\log N - 1)$ stage MINs.

4. On the equivalence of $\log N$ Stage N-MINs

In this section we deal with $\log N$ stage N-MINs, therefore—where no confusion arises—whenever we speak about N-MINs we mean $\log N$ stage N-MINs.

Bermond et al. [4] give an interesting characterization of N-MINs topologically equivalent to the Reverse Baseline network. It is based on the Banyan property (cf. Definition 2.4) and on the $P(*,*)$ property, that we briefly remind here.

Property $P(i,j)$. An N-MIN has property $P(i,j)$ for $1 \leqslant i \leqslant j \leqslant \log N$ if the subgraph $G_{i,j}$ induced by the nodes of the stage from i to j has exactly $2^{\log N - 1 - j + i}$ connected components.

Property $P(*,*)$. An N-MIN has property $P(*,*)$ if and only if it satisfies $P(i,j)$ for every ordered pair i,j such that $1 \leqslant i \leqslant j \leqslant \log N$.

Theorem 8 (Bermond et al. [4]). *All the N-MINs satisfying the Banyan Property and $P(*,*)$ are topologically equivalent to the Reverse Baseline.*

Checking the topological equivalence of an N-MIN to the Reverse Baseline using the characterization of Bermond, Fourneau and Jean-Marie requires $O(N^2 \log N)$ time, since the Banyan and $P(*,*)$ properties can be checked in $O(N^2 \log N)$ and $O(N \log^2 N)$ time, respectively. This time complexity has been improved by Hu et al. [14], who presented a simplified checking equivalence algorithm based on a marking scheme of nodes requiring $O(N \log N)$ time, that is optimal, since it is the same order of magnitude as the number of nodes in the N-MIN.

In this section, first we provide an alternative characterization for the equivalence class which the Reverse Baseline belongs to, then we describe an algorithm exploiting the characterization, checking the equivalence in $O(N \log N)$ time. Although the time complexity is not better than the previously known one, we detail this result for several reasons: first, it uses a different technique (i.e. LCP), second, it is much easier than the marking scheme in [14], finally it is preliminary to the algorithm described in the next section, where $(2\log N - 1)$ stage N-MINs are considered.

Lemma 9. *An N-MIN G satisfies the Banyan and $P(*,*)$ properties if and only if it can be decomposed as $\Delta \otimes \nabla$.*

Proof. ($\Leftarrow$) If G can be decomposed as $\Delta \otimes \nabla$ then G satisfies the Banyan property, in view of Lemma 5. G satisfies also $P(*,*)$; indeed, for any i,j such that $1 \leqslant i \leqslant j \leqslant \log N$, the subgraph of Δ induced by the nodes of the layers from i to j has exactly 2^{i-1} connected components and the subgraph of ∇ induced by the nodes of the layers from i to j has exactly $2^{\log N - j}$ connected components. From Lemma 2, the subgraph of G induced by the nodes of the stages from i to j has exactly $2^{\log N - j + i - 1}$ connected components, i.e. G satisfies $P(i,j)$ for any i,j and therefore G satisfies $P(*,*)$.

($\Rightarrow$) If G satisfies the Banyan and $P(*,*)$ properties, then—in view of Theorem 8—G is topologically equivalent to the Reverse Baseline. It is well known that the Reverse Baseline is topologically equivalent to the Butterfly network [19], which can be decomposed as LCP of $\Delta \otimes \nabla$, as stated in Lemma 4. Theorem 6 ensures that also the Reverse Baseline can be decomposed as LCP of $\Delta \otimes \nabla$ and, consequently G is the LCP of the same factors. □

Theorem 10. *An N-MIN G is decomposable as $\Delta \otimes \nabla$ if and only if G is topologically equivalent to the Reverse Baseline.*

Proof. ($\Rightarrow$) From the *if* part of Lemma 9 and Theorem 8, the statement follows.

($\Leftarrow$) As the Reverse Baseline satisfies the Banyan and the $P(*,*)$ properties, from the *only if* part of Lemma 9, the Reverse Baseline can be decomposed as $\Delta \otimes \nabla$, and from the *if* part of Theorem 6, the statement follows. □

Remark 1. As a consequence of this theorem and of the known equivalence of Butterfly, Omega, Flip, Reverse Baseline, etc. we deduce that these networks can all be decomposed as $\Delta \otimes \nabla$, as shown in Fig. 6. Furthermore, since (i) the LCP is commutative, (ii) the reverse operator is distributive, and (iii) the reverse of a Δ is a ∇ (and vice versa), it follows that each N-MIN decomposable as $\Delta \otimes \nabla$ is equivalent to its reverse.

In the following we describe the algorithm for the decomposition of a $\log N$ stage MIN G as $\Delta \otimes \nabla$; the algorithm assigns a label to nodes of G: the label of node v is a pair (v', v''), where v' and v'' represent the factors of v according to its decomposition.

The algorithm consists of two phases: the first one assigns the first elements of the labels, corresponding to labels of factor Δ, the second phase assigns the second elements of the labels corresponding to labels of factor ∇. If the algorithm succeeds in completing the labeling, then it provides the decomposition as $\Delta \otimes \nabla$; if, during the execution, two different labels are assigned to the same node, then the algorithm returns an error message and the N-MIN cannot be decomposed as $\Delta \otimes \nabla$, i.e. it is not topologically equivalent to the Reverse Baseline.

Let $d_1, d_2, \ldots, d_{N-1}$ and $n_1, n_2, \ldots, n_{N-1}$ be the names of the nodes in Δ and ∇, respectively. As usual, let nodes d_{2i} (n_{2i}) and d_{2i+1} (n_{2i+1}) be the children of d_i (n_i)—see Figs. 6a and b.

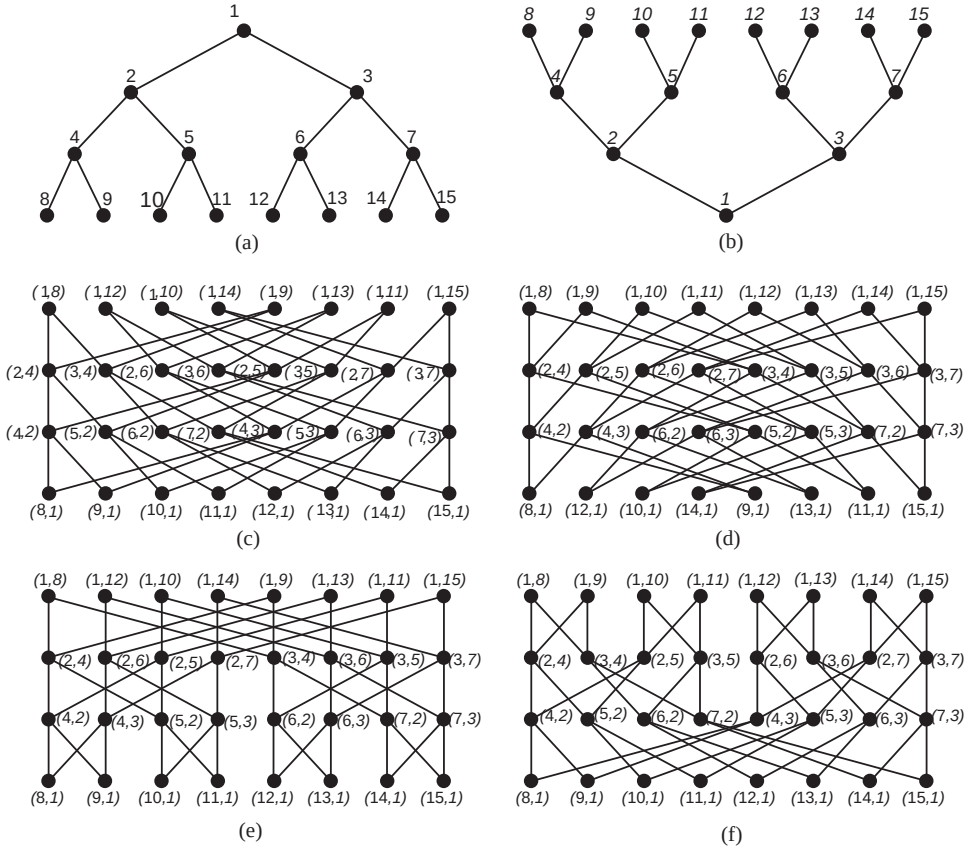

Fig. 6. (a) a Δ; (b) a ∇; (c) an Omega network; (d) a Flip network; (e) a Butterfly network; (f) a Reverse Baseline Network. All networks from (c)–(f) are the LCP of Δ and ∇, as highlighted by the labels, and are all topologically equivalent.

Algorithm $\Delta\otimes\nabla$-Decomposition
Input: a $\log N$ Stage N-MIN G;
Output: a labeling of nodes of G representing the decomposition as $\Delta\otimes\nabla$, if it exists; an error message otherwise.

a. **Assignment of the first element of the labels**

1. Consider any node v at the first stage of G and assign d_1 to it, corresponding to the root of Δ.
2. Recursively consider a node already labeled with d_i and assign labels d_{2i} and d_{2i+1} to its (not labeled) adjacent nodes at next stage, until the last stage of the N-MIN is reached. If there exists a node receiving a label more than once, exit and return an error message—see Fig. 7a.
3. For each stage s from $\log N$ to 2 do
   Repeat
   - Consider each node v at stage s labeled d_i, and assign the same label $d_{\lfloor i/2\rfloor}$ to its adjacent nodes at stage $s-1$.
   - Exit and return an error message if at least one of them is labeled:
     either differently by $d_{\lfloor i/2\rfloor}$
     or with $d_{\lfloor i/2\rfloor}$ and the sibling of v is not labeled d_{i+1} (d_{i-1}) if i is even (odd)—see Fig. 7b.

   Until all nodes at stage s are labeled or the algorithm returns an error.

b. **Assignment of the second element of the labels**

Assign the second elements of the labels, corresponding to labels of factor ∇, proceeding exactly as in phase a, but reversing the order of stages; namely, start from the last stage at Step 1, go up to the first one running Step 2, and down again by means of Step 3. The role of d_i is now done by n_i.

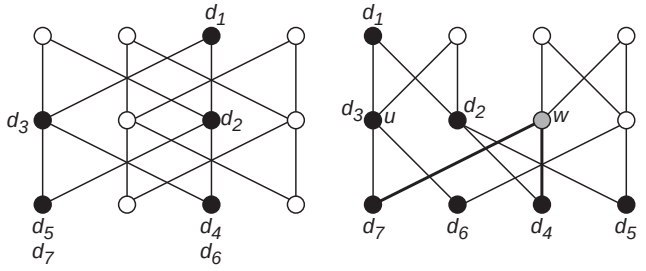

Fig. 7. Examples showing how algorithm $\Delta\otimes\nabla$-Decomposition fails during Step a.2 and a.3, respectively. White nodes are still unlabeled.

Consider the assignment of the first element of the labels (for the second element similar considerations hold): Steps a.1 and a.2 of algorithm $\Delta\otimes\nabla$-Decomposition assign the labels to a set of nodes inducing a complete binary tree and, in particular, label all nodes at the last stage of G. If this is not true, then G cannot be decomposable as $\Delta\otimes\nabla$, since it neither contains a complete binary tree rooted at the first considered node (see Fig. 7a). Step a.3 starts from the last stage and goes up labeling all other unlabeled nodes. If the labeling is computed without inconsistencies, a decomposition in factors is provided, otherwise the algorithm exits returning an error message. Note that a correct labeling requires that, during Step a.3, either node v labeled d_i labels unlabeled nodes, or it visits nodes already labeled $d_{\lfloor i/2\rfloor}$; in both cases it checks if v' sibling is labeled d_{i+1} when i even and d_{i-1} when i odd (see Fig. 7b).

Example. Fig. 8 shows how algorithm $\Delta\otimes\nabla$-Decomposition works on a MIN decomposable as $\Delta\otimes\nabla$. Namely, in Fig. 8a the first element of the labels of a complete binary tree, obtained running Steps a.1 and a.2, are shown. Fig. 8b highlights a single iteration of Step a.3: let v be labeled d_5; it assigns label $d_{\lfloor 5/2\rfloor}=d_2$ to unlabeled node w and finds node u already labeled with the same label; v' sibling is correctly labeled d_4. Finally, Fig. 8c illustrates the output of the algorithm.

The following theorem proves the correctness of algorithm $\Delta\otimes\nabla$-Decomposition and computes its time complexity.

Theorem 11. *The topological equivalence between a given N-MIN G and the Reverse Baseline network is checkable in $O(N\log N)$ time.*

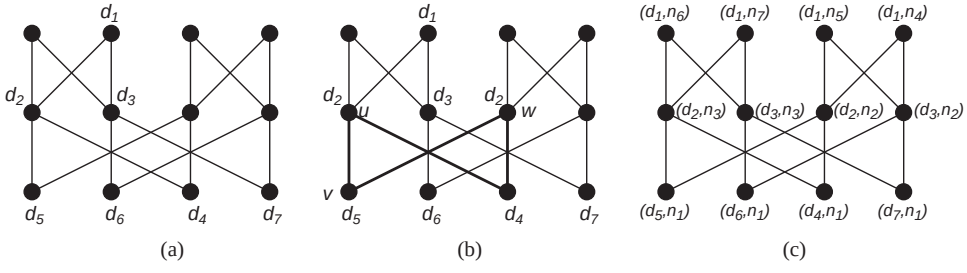

Fig. 8. An example showing how algorithm $\Delta\otimes\nabla$-Decomposition works.

Proof. In view of Theorem 10, it is enough to prove the correctness of algorithm $\Delta\otimes\nabla$-Decomposition, and to compute its time complexity.

Correctness. To multiply Δ and ∇ by means of LCP is equivalent to multiply Δ and a set of $\frac{N}{2}$ simple paths going from each leaf to the root, whose union is ∇. In view of Fact 1, in G we have to individuate $\frac{N}{2}$ different complete binary trees, each one having a different root at the first stage and all sharing the leaves at the last stage. All these trees, images of factor Δ, must be connected each other analogously to their second factors, that are two $\log N$ long simple paths. Phase a of the algorithm looks for all $\frac{N}{2}$ images of Δ in G. It is not difficult to see that the conditions leading to return an error in algorithm $\Delta\otimes\nabla$-Decomposition are necessary and sufficient for the existence of such trees. Similar considerations hold for Phase b.

Time complexity. The algorithm processes each node a constant number of times, and therefore the running time is linear in the number of nodes of G, i.e. $O(N\log N)$. Furthermore, the conditions to return an error are easily checkable in constant time by comparing some labels. □

5. On the equivalence of $(2\log N-1)$ stage networks

A $(2\log N-1)$ stage N-MIN is classically obtained as concatenation of two N-MINs with $\log N$ stages each, by merging the last stage of the first one with the first stage of the second one. It is typical to concatenate all the combinations of pairs of networks among Butterfly, Omega, Flip, Baseline, their reverses, etc. to obtain a new N-MIN. In the following we call N-MIN^2 a network G with $(2\log N-1)$ stages obtained concatenating two $\log N$ stage N-MINs, both equivalent to the Reverse Baseline.

In this section we deal with the equivalence problem for N-MIN^{2}s. Hu et al. [14] provide an algorithm to check the equivalence of two N-MIN^{2}s in $O(N^4\log N)$ time. We improve this result by solving the same problem in $O(N\log N)$ time, that is optimal.

In view of the considerations done in the previous section, both the $\log N$ stage N-MINs constituting a given N-MIN^2 can be decomposed as LCP of $\Delta\otimes\nabla$. As a consequence, we obtain that the factors of G are the concatenation of a Δ and a ∇ (roots merging) and of a ∇ and a Δ (leaves merging), respectively. It is obvious how

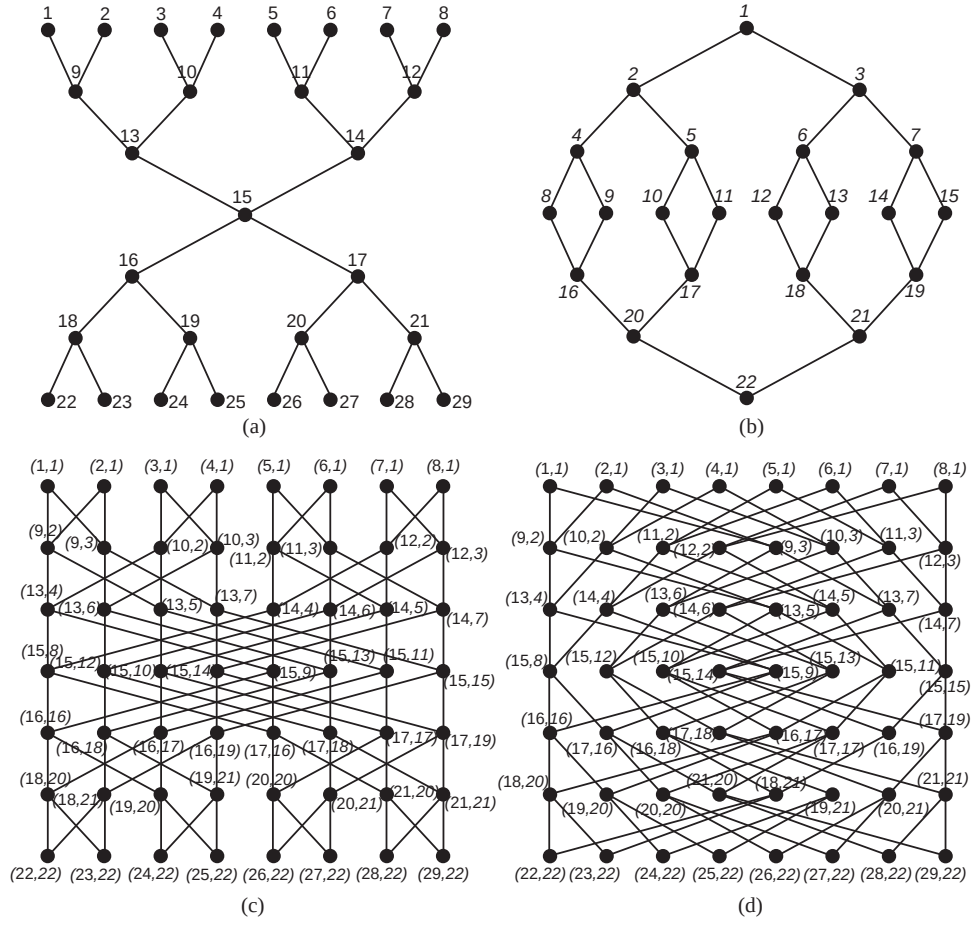

Fig. 9. (a) a $\frac{\nabla}{\Delta}$; (b) a $\frac{\Delta}{\nabla}$; (c) the concatenation of a reverse Butterfly and a Butterfly; (d) the concatenation of a Flip and an Omega. The networks in (c) and (d) are equivalent.

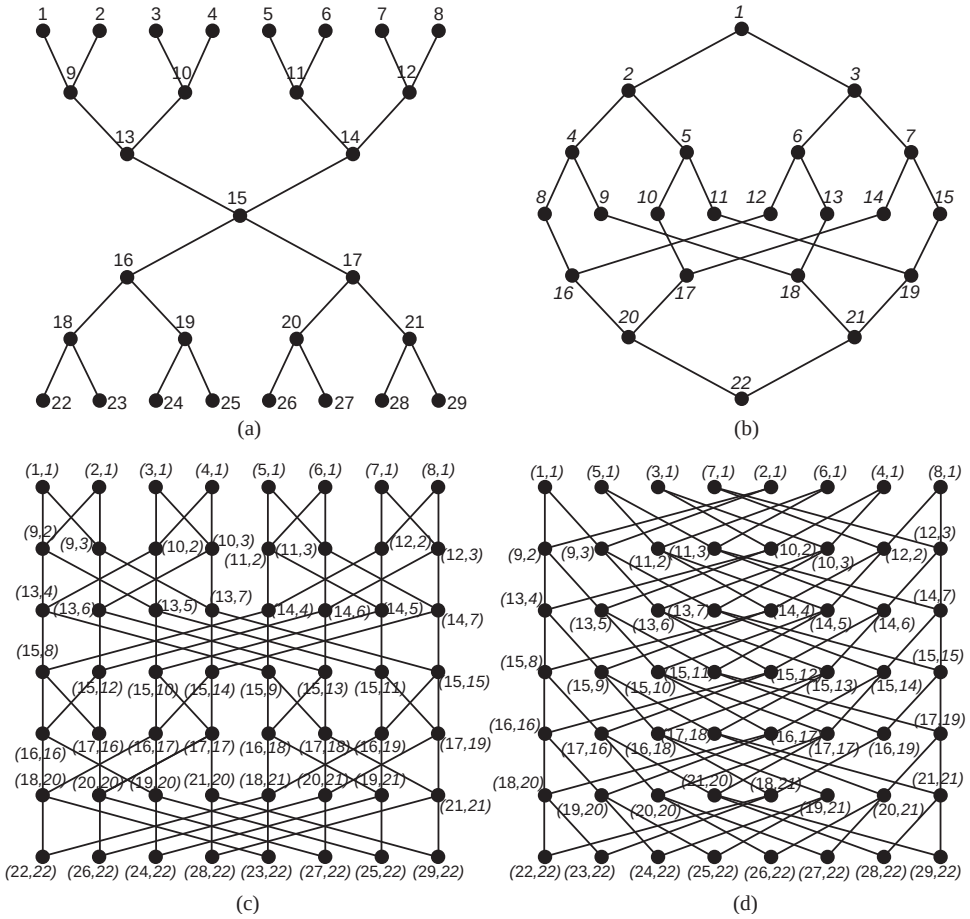

Fig. 10. (a) a $\begin{smallmatrix}\nabla\\ \Delta\end{smallmatrix}$; (b) a $\begin{smallmatrix}\Delta\\ \nabla\end{smallmatrix}$; (c) the concatenation of two reverse Butterflies; (d) the concatenation of two Omega networks. The networks in (c) and (d) are equivalent.

to merge the last layer of a ∇ with the first layer of a Δ, but there are many ways of merging the last layer of a Δ and the first layer of a ∇. In fact, from non equivalent MIN^{2}s different ways of connecting the leaves of the two trees derive, as highlighted in Figs. 9 and 10.

In the following, we indicate by $\begin{smallmatrix}\nabla\\ \Delta\end{smallmatrix}$ the concatenation of a ∇ and a Δ; we indicate by $\begin{smallmatrix}\Delta\\ \nabla\end{smallmatrix}$ any concatenation of a Δ and a ∇; among all the possible $\begin{smallmatrix}\Delta\\ \nabla\end{smallmatrix}$s we distinguish the ordered concatenation (Fig. 9b) by calling it *obvious* $\begin{smallmatrix}\Delta\\ \nabla\end{smallmatrix}$.

Remark 2. All N-MIN^{2}s, obtained as concatenation of any $\log N$ stage N-MIN topologically equivalent to the Reverse Baseline and its reverse, are topologically equivalent and their $\begin{smallmatrix}\Delta\\ \nabla\end{smallmatrix}$ factor is an obvious $\begin{smallmatrix}\Delta\\ \nabla\end{smallmatrix}$.

Intuitively, the previous statement follows from the fact that the N-MIN^2 obtained by the concatenation of a network and its reverse is symmetric with respect to the line passing through the conjunction stage, then the concatenation is obtained by identifying nodes with the same labels (see Fig. 9). This allows one to characterize the equivalence class of all N-MIN^{2}s obtained by any Reverse Baseline equivalent N-MIN and its reverse: it contains exactly all N-MIN^{2}s decomposed as the LCP of a $\begin{smallmatrix}\nabla\\ \Delta\end{smallmatrix}$ and an obvious $\begin{smallmatrix}\Delta\\ \nabla\end{smallmatrix}$.

Now we describe a new efficient algorithm that, given in input two N-MIN^{2}s, verifies whether they are topologically equivalent or not.

The algorithm is divided into two parts: the first one consists in decomposing each N-MIN^2 into a $\begin{smallmatrix}\nabla\\ \Delta\end{smallmatrix}$ and a $\begin{smallmatrix}\Delta\\ \nabla\end{smallmatrix}$; the second one decomposes $\begin{smallmatrix}\Delta\\ \nabla\end{smallmatrix}$ factors into prime factors and then compare them pairwise: if they are topologically equivalent, then the N-MIN^{2}s are, too in view of Theorem 6.

For clarity of explanation, we will deal with the two parts separately.

5.1. Decomposing G as $\begin{smallmatrix}\nabla\\ \Delta\end{smallmatrix}\otimes\begin{smallmatrix}\Delta\\ \nabla\end{smallmatrix}$

Consider an N-MIN^2 G. For the previous reasonings, G is always decomposable as $\begin{smallmatrix}\nabla\\ \Delta\end{smallmatrix}\otimes\begin{smallmatrix}\Delta\\ \nabla\end{smallmatrix}$. The aim of the following algorithm is two-fold: first, it discloses this decomposition by labeling each node v of G with a pair (v', v''), where v' and v'' represent the factors of v according to its decomposition; second, it completely defines the second factor $\begin{smallmatrix}\Delta\\ \nabla\end{smallmatrix}$, i.e. the way of connecting

nodes at stage $\log N$ and at stage $\log N + 1$ of $\begin{smallmatrix}\Delta\\\nabla\end{smallmatrix}$. The algorithm is divided into three different phases.

The first one decomposes the first $\log N$ stages of the N-MIN^2 using algorithm $\Delta\otimes\nabla$-Decomposition.

The second phase decomposes the last $\log N - 1$ stages. To this end, observe that—since G has the $P(*,*)$ property—the last $\log N - 1$ stages of G induce two connected components, whose factors are a subgraph ∇' of $\begin{smallmatrix}\Delta\\\nabla\end{smallmatrix}$ and a pair Δ'_1 and Δ'_2, subgraphs of $\begin{smallmatrix}\nabla\\\Delta\end{smallmatrix}$. This second phase labels one of the two connected components by means of algorithm $\Delta\otimes\nabla$-Decomposition, starting from an arbitrary node. In order to guarantee the consistency in the labeling, it is not possible to start from an arbitrary node to label the second connected component: each unlabeled node at stage $\log N + 1$ of G receives the same second element of the label (deriving from ∇') as its labeled sibling (with respect to their father at stage $\log N$) in the first connected component. Once nodes at stage $\log N + 1$ have been labeled, we can run algorithm $\Delta\otimes\nabla$-Decomposition, in order to label the second component.

At the end of these two phases all nodes of G have been labeled, and hence decomposed as LCP. Also all edges, but the edges between stages $\log N$ and $\log N + 1$, are decomposed as LCP. The third phase adds the edges between layers $\log N$ and $\log N + 1$ of $\begin{smallmatrix}\Delta\\\nabla\end{smallmatrix}$. In fact, the decomposition of stages $\log N$ and $\log N + 1$ of G is completely specified by means of labels. Exploiting the edge-stage of G in between and the labels assigned during the previous phases, the corresponding edge-stage in $\begin{smallmatrix}\Delta\\\nabla\end{smallmatrix}$ is built. Namely, an edge in $\begin{smallmatrix}\Delta\\\nabla\end{smallmatrix}$ is added between nodes u and v if an edge in the N-MIN exists between nodes having as second element of their label u and v, respectively. It is always possible to perform this third phase, in view of the definition of G as concatenation of two $\log N$ Reverse Baseline equivalent MINs.

The pseudo-code of the just described algorithm follows.

Algorithm $\begin{smallmatrix}\nabla\\\Delta\end{smallmatrix}\otimes\begin{smallmatrix}\Delta\\\nabla\end{smallmatrix}$-Decomposition
Input: a $(2\log N - 1)$ Stage N-MIN^2 G;
Output: a specific $\begin{smallmatrix}\Delta\\\nabla\end{smallmatrix}$, and a labeling of nodes of G representing the decomposition $G = \begin{smallmatrix}\nabla\\\Delta\end{smallmatrix}\otimes\begin{smallmatrix}\Delta\\\nabla\end{smallmatrix}$.

a. **Decomposition of the first $\log N$ stages of the N-MIN^2**
   Run algorithm $\Delta\otimes\nabla$-Decomposition on the first $\log N$ stages of the N-MIN^2 and produce a label for each node of these stages.
b. **Decomposition of the last $\log N - 1$ stages**
   1. Label one of the two connected components by running algorithm $\Delta\otimes\nabla$-Decomposition, starting from an arbitrary node.
   2. Consider the second element of the label. Label each unlabeled node at stage $\log N + 1$ of G with the same label as its labeled sibling (with respect to their father at stage $\log N$).
   3. Run algorithm $\Delta\otimes\nabla$-Decomposition on the last $\log N - 1$ stages of the N-MIN^2, taking into account the labels assigned at stage $\log N + 1$.
c. **Decomposition of the set of edges between stages $\log N$ and $\log N + 1$**
   For each edge $e = (u, v)$ connecting node u—labeled (u', u'')—at stage $\log N$ and node v—labeled (v', v'')—at stage $\log N + 1$ of G do
   add edge (u'', v'') in $\begin{smallmatrix}\Delta\\\nabla\end{smallmatrix}$ between node u'' at stage $\log N$ and node v'' at stage $\log N + 1$.

Before presenting an example showing how algorithm $\begin{smallmatrix}\nabla\\\Delta\end{smallmatrix}\otimes\begin{smallmatrix}\Delta\\\nabla\end{smallmatrix}$-Decomposition works, for the sake of clearness we add some details omitted in the pseudocode. First observe that phase a must decompose the first $\log N$ stages of G as LCP of a ∇ (first factor) and a Δ (second factor); hence, the output of algorithm $\Delta\otimes\nabla$-Decomposition executed during this phase must be slightly modified: actually, the pairs of labels must be inverted.

Notice also that nodes of $\begin{smallmatrix}\nabla\\\Delta\end{smallmatrix}$ and $\begin{smallmatrix}\Delta\\\nabla\end{smallmatrix}$ not always follow the rule that node i has children $2i$ and $2i + 1$, as required by algorithm $\Delta\otimes\nabla$-Decomposition, but it is sufficient to slightly modify the code in order to allow it to be run on opportune portions of $\begin{smallmatrix}\nabla\\\Delta\end{smallmatrix}$ and $\begin{smallmatrix}\Delta\\\nabla\end{smallmatrix}$.

Example. Consider Fig. 11a: it shows an N-MIN^2 and a decomposition in factors of the first $\log N$ stages performed by algorithm $\Delta\otimes\nabla$-Decomposition (with the modifications just mentioned). In Fig. 11b the last $\log N - 1$ stages are factorized, and the two connected components are highlighted. Observe that first one of the components is completely labeled (e.g. the dotted one), then the labels of nodes at stage $\log N - 1$ of the second component are set according to the labels of the first component and of edge-stage between stages $\log N$ and $\log N + 1$. Finally, Fig. 11c shows the $\begin{smallmatrix}\Delta\\\nabla\end{smallmatrix}$ factor after phase c has been executed.

Theorem 12. *Given an N-MIN^2 G, $O(N\log N)$ time is sufficient to compute the specific $\begin{smallmatrix}\Delta\\\nabla\end{smallmatrix}$ deriving from the decomposition of G as $\begin{smallmatrix}\nabla\\\Delta\end{smallmatrix}\otimes\begin{smallmatrix}\Delta\\\nabla\end{smallmatrix}$.*

Proof. The statement is proved if we show that algorithm $\begin{smallmatrix}\nabla\\\Delta\end{smallmatrix}\otimes\begin{smallmatrix}\Delta\\\nabla\end{smallmatrix}$-Decomposition is correct and that its time complexity is $O(N\log N)$. To this aim, we consider the algorithm phase by phase.

Correctness. In view of the definition of the N-MIN^2, Phase a of the algorithm never fails and always returns a labeling.

Also phase b always succeeds, and the consistency of the labels is guaranteed from the fact that the starting labels of the second connected component are not

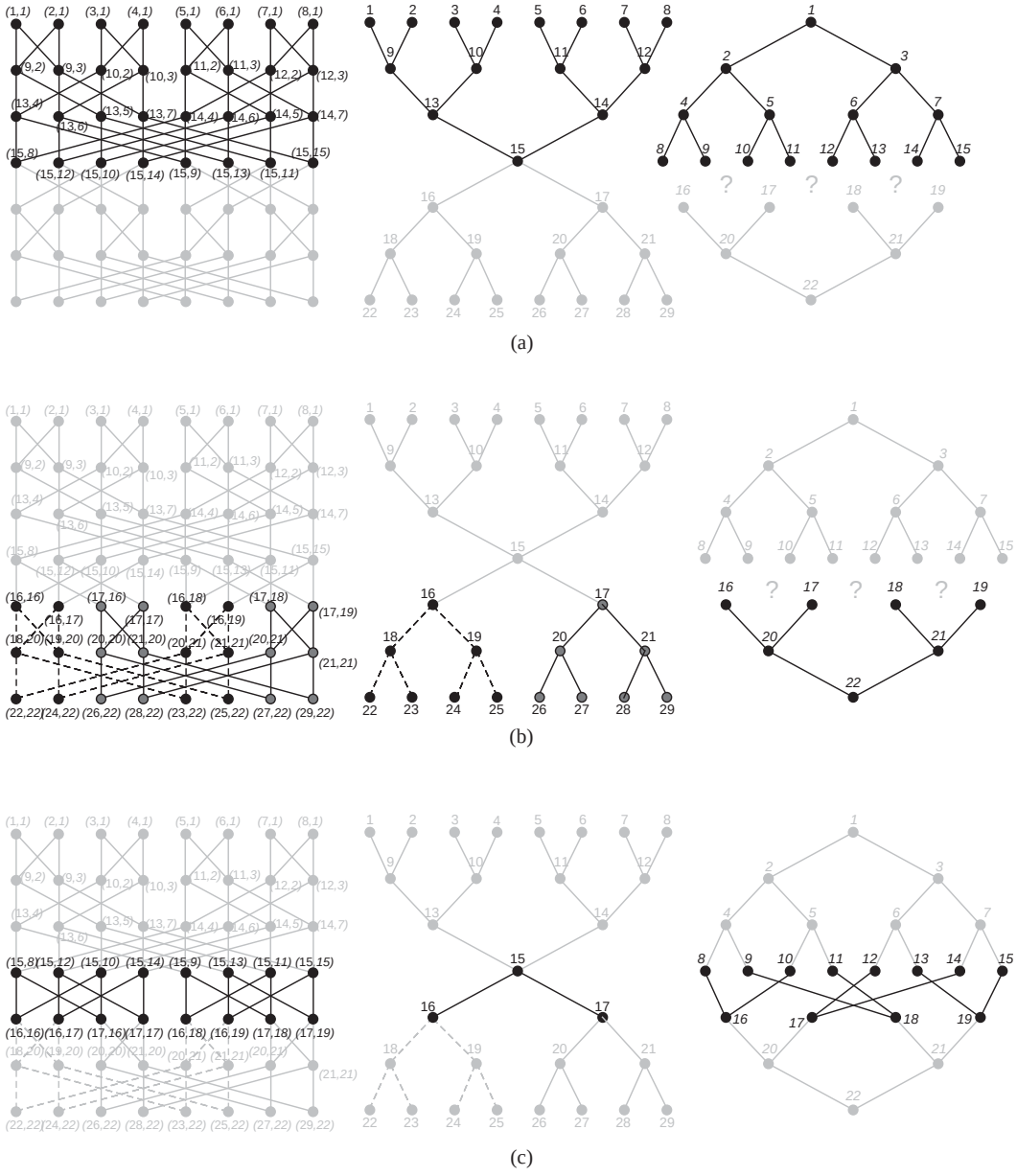

Fig. 11. An example showing how algorithm $\frac{\nabla}{\Delta} \otimes \frac{\Delta}{\nabla}$-Decomposition works.

arbitrary, but are led by the labeling of the first component.

At the end of these two phases all nodes of G have been labeled, and hence decomposed as LCP. During phase c the only edge-stage characterizing each $\frac{\Delta}{\nabla}$ (and hence each N-MIN^2) is built, exploiting the labels representing the decomposition in factors.

From the proof of Theorem 11, from the definition of G and from the definition of LCP, the correctness of the algorithm follows.

Time complexity. The first two phases run in $O(N \log N)$ time (cf. Theorem 11); the third phase considers all edges between stages $\log N$ and $\log N + 1$ of G once, and therefore runs in $O(N)$ time. □

5.2. *Decomposing $\frac{\Delta}{\nabla}$ into prime factors*

In view of Theorem 6, checking the equivalence of two N-MIN^{2}s can be reduced to the simpler problem of checking the equivalence of their factors, if they have been opportunely constructed. To this aim, we further decompose the $\frac{\Delta}{\nabla}$s into their prime factors and then we compare them pairwise.

In Section 2, we have shown the decomposition of Δ and ∇ in terms of prime factors, i.e. $\Delta = \Lambda_1 \otimes \Lambda_2 \otimes \cdots \otimes \Lambda_{\log N-1}$ and $\nabla = V_1 \otimes V_2 \otimes \cdots \otimes V_{\log N-1}$ (see Λ_i and V_i in Fig. 3).

We can decompose any $\frac{\Delta}{\nabla}$ as the LCP of $\log N - 1$ $\Phi_{i,j}$ (see $\Phi_{i,j}$ in Fig. 3), where the values of i and j depend on

the specific $\overset{\triangle}{\triangledown}$, and i and j must have the following properties:

- $1 \leqslant i \leqslant \log N - 1$ and $\log N \leqslant j \leqslant 2 \log N - 2$;
- for any pair of factors $\Phi_{i,j}$ and $\Phi_{i',j'}$, it must be $i \neq i'$ and $j \neq j'$, i.e. each fixed value of $i(j)$ appears exactly once.

Two examples of decomposition of $\overset{\triangle}{\triangledown}$s into prime factors are shown in Fig. 12 in the case $N = 8$.

A possible way to efficiently determine $\Phi_{i,j}$ factors of $\overset{\triangle}{\triangledown}$ consists in exploiting simple cycles from i to j inside $\overset{\triangle}{\triangledown}$. More precisely, starting from any node v of the upper part of $\overset{\triangle}{\triangledown}$ at layer i, it is easy to look for the value $\tilde{\jmath}$ such that there exists a simple cycle from i to $\tilde{\jmath}$ passing through v. In view of Lemma 3, this implies that $\Phi_{i,\tilde{\jmath}}$ is a factor of $\overset{\triangle}{\triangledown}$. This operation takes $O(N)$ time as we have to search possibly the whole $\overset{\triangle}{\triangledown}$.

All $\Phi_{i,j}$ factors can be found repeating the previous search once for each $i = 1, \ldots, \log N - 1$. Indeed, the simplicity of $\Phi_{i,j}$ factors implies that if $\Phi_{i,\tilde{\jmath}}$ is a factor of $\overset{\triangle}{\triangledown}$, then each node at layer i in $\overset{\triangle}{\triangledown}$ belongs to a simple cycle from i to $\tilde{\jmath}$.

It follows that the whole decomposition of $\overset{\triangle}{\triangledown}$ into prime factors can be done in $O(N \log N)$ time.

Now, we are ready to detail the whole algorithm checking the equivalence of two N-MIN^{2}s.

Algorithm Compare-N-MIN^{2}s

Input: two $(2 \log N - 1)$ layer N-MIN^{2}s G' and G'';

Output: 1 if the N-MIN^{2}s are topologically equivalent; 0 otherwise.

a. **Decomposing N-MIN^{2}s**
   Run algorithm $\overset{\triangle}{\triangledown} \otimes \underset{\triangle}{\triangledown}$-`Decomposition` on G' and produce $\overset{\triangle}{\triangledown}'$;
   Run algorithm $\overset{\triangle}{\triangledown} \otimes \underset{\triangle}{\triangledown}$-`Decomposition` on G'' and produce $\overset{\triangle}{\triangledown}''$;

b. **Decomposing $\overset{\triangle}{\triangledown}$s into prime factors**
   Run procedure decomposing $\overset{\triangle}{\triangledown}$s into $\Phi_{i,j}$ factors both on $\overset{\triangle}{\triangledown}'$ and on $\overset{\triangle}{\triangledown}''$;

c. **Comparing $\Phi_{i,j}$ *factors***
   Sort $\Phi_{i,j}$ factors of $\overset{\triangle}{\triangledown}'$ according to increasing index i producing the ordered list $\langle j'_1, j'_2, \ldots, j'_{\log N - 1} \rangle$;
   Do the same for $\overset{\triangle}{\triangledown}''$ producing $\langle j''_1, j''_2, \ldots, j''_{\log N - 1} \rangle$;
   For $i = 1$ to $\log N - 1$ do
      if $(j'_i \neq j''_i)$
      then return 0;
   return 1.

Example. Consider the two $\overset{\triangle}{\triangledown}$s depicted in Fig. 13b as outputs of Phase a of algorithm `Compare` N-MIN^{2}s. Phase b looks for simple cycles from each i to some j (see Fig. 13b) decomposing both $\overset{\triangle}{\triangledown}$s into their prime factors,

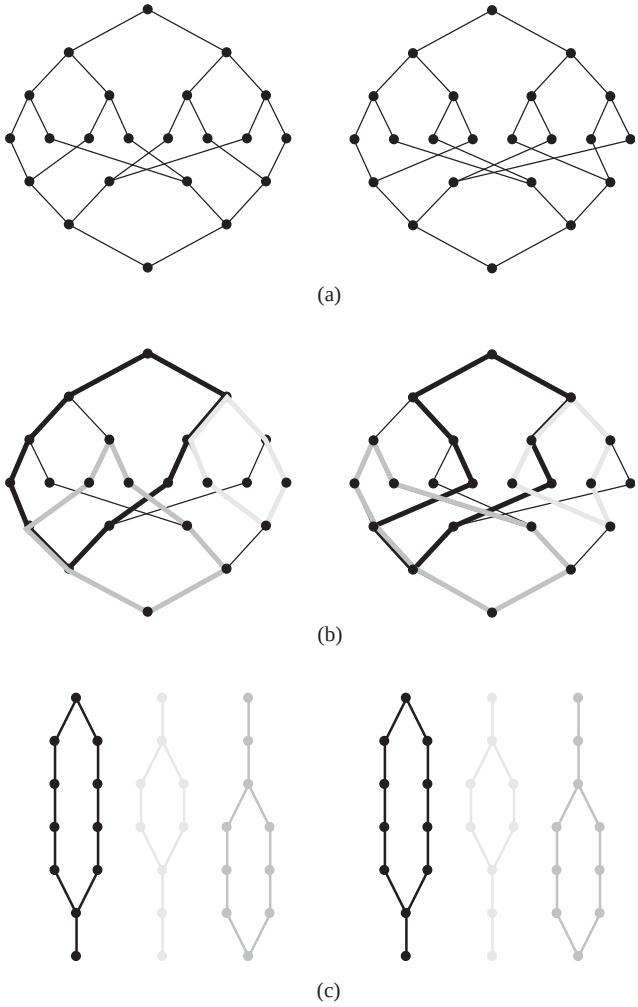

Fig. 13. (a) Two equivalent $\overset{\triangle}{\triangledown}$s, with *dn*-labels and labels on their layers; (b) performing the algorithm looking for the simple cycles and hence $\Phi_{i,j}$ factors; (c) $\Phi_{i,j}$ factors of the given $\overset{\triangle}{\triangledown}$s.

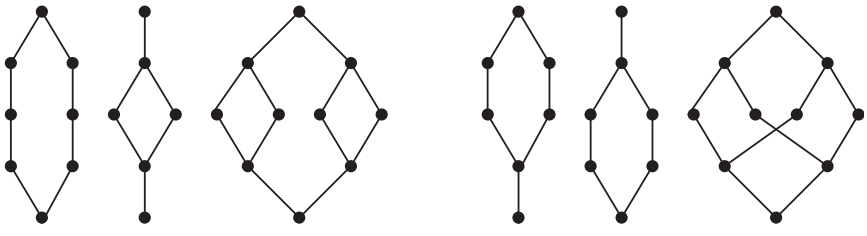

Fig. 12. Two $\overset{\triangle}{\triangledown}$s with $N = 8$ decomposed into prime factors.

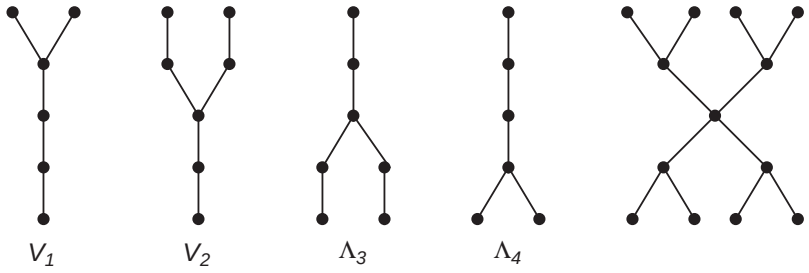

Fig. 14. A $\bigtriangledown\!\!\!\bigtriangleup$ with $N = 8$ decomposed into prime factors.

depicted in Fig. 13c. From them, Phase c deduces that the input N-MIN^{2}s are topologically equivalent.

Theorem 13. *Given two N-MIN^{2}s, $O(N \log N)$ time is sufficient to check their topological equivalence.*

Proof. We prove the assertion by showing that algorithm `Compare-`N-MIN^{2}s is correct and it runs in $O(N \log N)$ time.

Correctness. It directly descends from Theorem 6, from the associativity of LCP, and from all previous correctness proofs.

Time complexity. It is $O(N \log N)$ since both Phases a and b take $O(N \log N)$ time, and Phase c takes time proportional to the length of the sequences, i.e. $O(\log N)$. □

5.3. Characterization of equivalence class

We conclude this section providing a characterization of equivalence classes of N-MIN^{2}s in terms of prime factors and Butterfly-like stages, and we compute the number of equivalence classes. We observe that $\bigtriangledown\!\!\!\bigtriangleup = V_1 \otimes \cdots \otimes V_{\log N-1} \otimes \Lambda_{\log N} \otimes \cdots \otimes \Lambda_{2 \log N-2}$ (see Fig. 14 in the case $N = 8$). Reminding the decomposition into prime factors of $\bigtriangleup\!\!\!\bigtriangledown$, it follows that any N-MIN^2 G is decomposable into prime factors as LCP of $V_1 \otimes V_2 \otimes \cdots \otimes V_{\log N-1} \otimes \Lambda_{\log N} \otimes \Lambda_{\log N+1} \otimes \cdots \otimes \Lambda_{2 \log N-2} \otimes \Phi_{i_1,j_1} \otimes \cdots \otimes \Phi_{i_{\log N-1},j_{\log N-1}}$, where values of indices of $\Phi_{i,j}$ factors depend on the specific N-MIN^2, that is on the specific $\bigtriangleup\!\!\!\bigtriangledown$.

Since the LCP is commutative and associative, we can re-sort the prime factors according to their indices and group them into triples $\Phi_{i,j} \otimes V_i \otimes \Lambda_j = X_{i,j}$ (see Fig. 4).

The following theorem holds:

Theorem 14. *Each N-MIN^2 is the LCP of exactly $(\log N - 1) X_{i,j}$s, where the indices of each $X_{i,j}$ have the following properties*:

a. for each fixed $\tilde{\imath}$, $1 \leqslant \tilde{\imath} \leqslant \log N - 1$, there exists exactly one $X_{\tilde{\imath},j}$;

b. for each fixed $\tilde{\jmath}$, $\log N \leqslant \tilde{\jmath} \leqslant 2 \log N - 2$, there exists exactly one $X_{i,\tilde{\jmath}}$.

From the previous reasonings we are able to decompose any N-MIN^2 into its $X_{i,j}$ factors in $O(N \log N)$ time. Now we specify how each $X_{i,j}$ factor contributes on the representation of the resulting G, in terms of Butterfly-like edge-stages.

We can consider the set of $X_{i,j}$s ordered with respect to their first index: $X_{\log N-1,j_{\log N-1}} \otimes X_{\log N-2,j_{\log N-2}} \otimes \cdots \otimes X_{2,j_2} \otimes X_{1,j_1}$. In this order, the general $X_{i,j}$ is the $(\log N - i)$th factor and, in correspondence of edge-stages i and j, it produces two ith Butterfly-like edge-stages, i.e. the edge-stages with crosses of width $2^i - 1$ (see Fig. 15). In other words, after ordinately multiplying these factors, we get a N-MIN^2 equivalent to G with the property that the upper Reverse Baseline equivalent network is a Reverse Butterfly, while the lower one is any permutation of all the possible Butterfly-like edge-stages. It is easy to see that different sequences of j indexes in the list of $X_{i,j}$ factors lead to non-equivalent N-MIN^{2}s; hence, we can choose these special N-MIN^{2}s as particular representatives of each equivalence class. Furthermore, since the number of possible permutations of the lower $(\log N - 1)$ edge-stages is $(\log N - 1)!$ we can also state the following theorem:

Theorem 15. *The number of distinct equivalence classes of N-MIN^{2}s is $(\log N - 1)!$*

It follows that also the number of different $\bigtriangleup\!\!\!\bigtriangledown$s must be $(\log N - 1)!$, as highlighted in Fig. 16.

It is straightforward to notice that, for each fixed N, the number of equivalence classes is different. This implies that any two N-MIN^{2}s belonging to different classes for a certain N can fall in the same class for a smaller N.

6. Conclusions and open problems

In this paper we have proposed a new general characterization of topological equivalence of N-MINs based on the LCP. Then, we have applied this characterization to $\log N$ and $(2 \log N - 1)$ stage N-MINs.

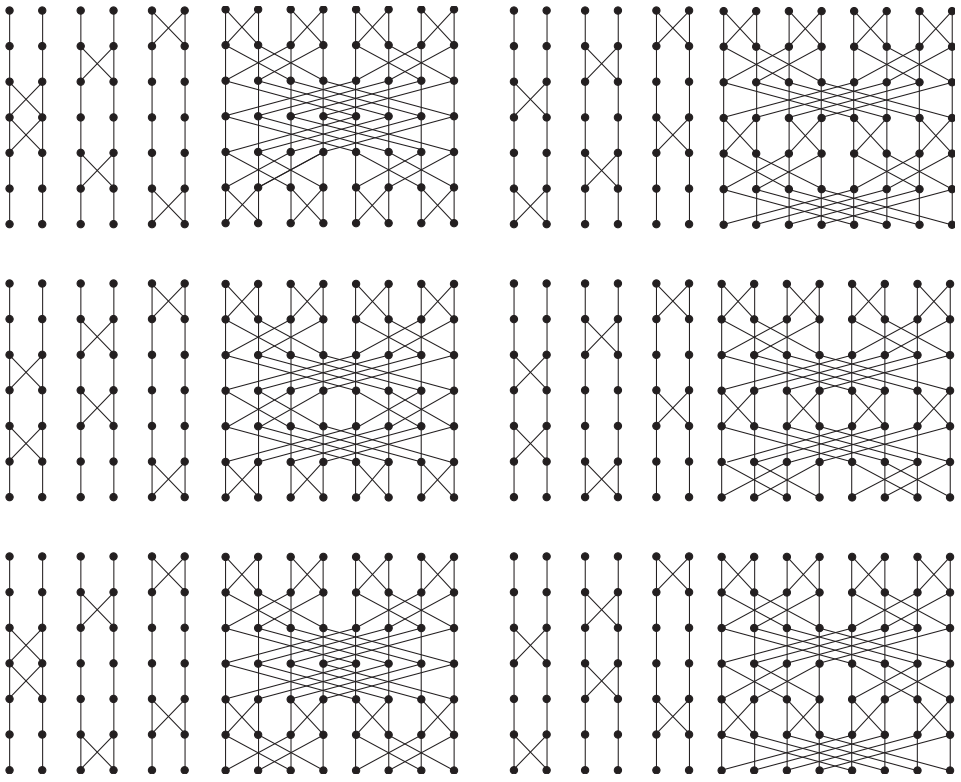

Fig. 15. All the possible ways of multiplying $X_{i,j}$s and the representative of the corresponding equivalence class obtained by using only Butterfly-like stages, for $N = 16$.

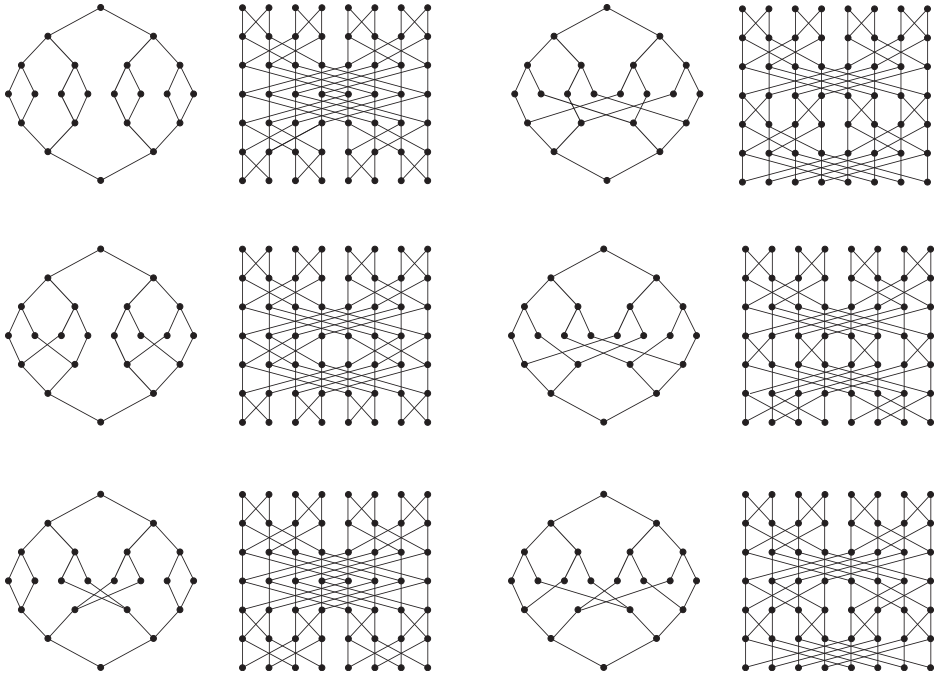

Fig. 16. All the possible $\frac{\Delta}{\nabla}$s related to $N = 16$ and a representative of the corresponding equivalence class.

Namely, we have designed an algorithm to determine if a given $\log N$ stage N-MIN is topologically equivalent to the Reverse Baseline. This algorithm is optimal, as other well known ones in the literature, but its interest lies in the novel approach to the problem.

About $(2\log N - 1)$ stage N-MINs (N-MIN^{2}s), we have provided an optimal algorithm testing the equivalence of two N-MIN^{2}s. This result improves of a factor N^3 the previously known one and definitely closes the equivalence problem on N-MIN^{2}s.

Observe that the results for $\log N$ and $(2\log N - 1)$ stage N-MINs are different, since the first one checks the belonging to the equivalence class whose the Reverse Baseline is a representative, while the second one checks the equivalence of any two N-MIN^{2}s. Nevertheless, it does not seem easy to cover this gap using the LCP, since it is not clear how to specify the properties of the complement of the equivalence class which the Reverse Baseline belongs to.

The technique used to check the equivalence of two N-MIN^{2}s has also led to determine the number of different equivalence classes and to characterize them. Namely, we have characterized each equivalence class by means of $X_{i,j}$ factors. Furthermore, we have proved that a representative of each equivalence class can be visualized as an N-input Reverse Butterfly concatenated with a $\log N$ stage N-MIN obtained as any permutation of the Butterfly-like edge-stages. This is the first result giving body to the equivalence classes not containing the most popular and studied N-MIN^{2}s and contributes to approach problems on N-MIN^{2}s independently from the graphical appearance of the specific network.

Finally, the characterization of the equivalence classes of N-MIN^{2}s allows one to do some considerations about the rearrangeability; this is the subject of a further paper [6].

Acknowledgments

We thank Shimon Even for his careful reading of the first draft of this work. His suggestions and comments have helped us to improve the soundness and the presentation of this paper. We thank also Adolfo Piperno: the interesting discussions concerning the equivalence of graphs with him have led us to design the algorithm checking the equivalence of two $\overset{\Delta}{\nabla}$s. Finally, we want to thank the anonymous referees, whose comments have substantially contributed to improve the paper.

References

[1] D.P. Agrawal, Graph theoretical analysis and design of multistage interconnection networks, IEEE Trans. Comput. C32 (1983) 637–648.

[2] K.E. Batcher, The flip network in STARAN, Proceedings of the International Conference on Parallel Processing, 1981, pp. 65–71.

[3] V.E. Beneš, On rearrangeable three-stage connecting networks, Bell System Tech. J. XLI (1962) 1481–1492.

[4] J.C. Bermond, J.M. Fourneau, A. Jean-Marie, Equivalence of multistage interconnection networks, Inform. Process. Lett. 26 (1987) 45–50.

[5] T. Calamoneri, M. Di Ianni, Interval routing & layered cross product: compact routing schemes for butterfly, mesh of trees, fat trees and Beneš networks, J. Parallel Distrib. Comput. (2003) in press; A preliminary version appeared in Proceedings of the Fourth International Euro-Par Conference, Lecture Notes in Computer Science, Vol. 1470, Springer, Berlin, 2001, pp. 1029–1039.

[6] T. Calamoneri, A. Massini, A new approach to the rearrangeability of $(2\log N - 1)$ stage MINs, Proceedings of IASTED International Symposium on Applied Informatics (AI 2001), Innsbruck, 2001, pp. 365–370.

[7] G.A. De Biase, Interconnection structures and parallel computing, in: D.J. Evans (Ed.), Advances in Parallel Computing, JAI Press, Vol. 1, 1991.

[8] J. Duato, S. Yalamanchili, L. Ni, Interconnection Networks: An Engineering Approach, IEEE Computer Society, Silver Spring, MD, 1997.

[9] G. Even, S. Even, Embedding interconnection networks in grid via the layered cross product, Networks 36 (2) (2000) 91–95.

[10] S. Even, A. Litman, Layered cross product—a technique to construct interconnection networks, Fourth ACM Symposium on Parallel Algorithms and Architectures (SPAA'92), San Diego, CA, 1992, pp. 60–69.

[11] T. Feng, Data manipulating functions in parallel processors and their implementations, IEEE Trans. Comput. C23 (1974) 309–318.

[12] T. Feng, Y. Kim, S. Seo, Interstage correlation in a class of multistage interconnection networks, Proceedings of the 1994 International Computer Symposium, 1994.

[13] M.R. Garey, D.S. Johnson, Computers and Intractability, A Guide to the Theory of NP-Completeness, W.H. Freeman and Co., San Francisco, 1979.

[14] Q. Hu, X. Shen, J. Yang, Topologies of combined $(2\log N - 1)$-stage interconnection networks, IEEE Trans. Comput. 46 (1997) 118–124.

[15] J. Köbler, U. Schöning, J. Torán, The Graph Isomorphism Problem, Its Structural Complexity, Birkhauser Boston Inc., Boston, MA, 1993.

[16] R. Kráľovič, B. Rovan, P. Ružička, Interval routing on layered cross product of trees and cycles, Proceedings of the Fifth International Euro-Par Conference, Lecture Notes in Computer Science, Vol. 1685, Springer, Berlin, 1999.

[17] D.H. Lawrie, Access and alignment of data in an array processor, IEEE Trans. Computers C24 (1975) 1145–1155.

[18] K.Y. Lee, On the rearrangeability of $2\log N - 1$-stage permutation networks, IEEE Trans. Comput. C34 (1985) 412–425.

[19] F.T. Leighton, Introduction to Parallel Algorithms and Architectures: Arrays, Trees and Hypercubes, Morgan Kaufmann Publications, San Mateo, CA, 1992.

[20] A. Paz, A theory of decomposition into prime factors of layered interconnection networks, manuscript, 2001. A preliminary version of this paper was presented at the Dagstuhl Seminar on Graphs Transformations in Computer Science, September 1996.

[21] M.C. Pease, The indirect binary cube microprocessor array, IEEE Trans. Comput. C26 (1977) 458–473.

[22] C. Wu, T. Feng, On a class of multistage interconnection networks, IEEE Trans. Comput. C29 (1980) 694–702.

[23] C. Wu, T. Feng, The reverse-exchange interconnection networks, IEEE Trans. Comput. C29 (1980) 801–811.

[24] Y. Yeh, T. Feng, On a class of rearrangeable networks, IEEE Trans. Comput. C41 (1992) 1361–1379.

Tiziana Calamoneri graduated in Mathematics in 1992 and got her Ph.D. in Computer Science in 1997, at University of Rome ''La Sapienza,'' Italy. Since 2000 she is assistant professor at the Department of Computer Science, University of Rome ''La Sapienza''.

Her research interests include parallel and sequential graph algorithms, two- and three-dimensional graph drawing, layout of networks topologies and optimal routing schemes, channel assigment in wireless networks.

Annalisa Massini graduated in Mathematics in 1989 and got her Ph.D. in Computer Science in 1993, at University of Rome ''La Sapienza,'' Italy. In 1996 she became assistant professor at the Department of Computer Science, University of Rome ''La Sapienza''. Since 2001 she is associate professor at the Department of Computer Science, University of Rome ''La Sapienza''.

Her research interests include algorithms on interconnection networks, layout of networks topologies, parallel arithmetic units.