A New Approach to the Rearrangeability of $(2\log N - 1)$ Stage MINs

Tiziana Calamoneri Annalisa Massini
Dip. di Scienze dell'Informazione
Università di Roma "La Sapienza", Italy
via Salaria 113 - 00198 Rome, Italy
{calamo,massini}@dsi.uniroma1.it

Abstract

In this paper we present a twofold result. First we provide a new routing algorithm to realize any permutation on all $(2\log N - 1)$ stage MINs in the class of Beneš equivalent networks. The time complexity is the same as the Looping algorithm, i.e. $O(N \log N)$, but the interest of the presented algorithm derives from that it is very general and it runs on every network in the class apart from its symmetry. Furthermore, it provides a constructive proof of rearrangeability for a wide class of networks, substituting all the different proofs presented in literature. Second, we prove the rearrangeability of a class of MINs whose rearrangeability was not known; for them we describe an $O(N \log N)$ time routing algorithm.

Keywords: Multistage Interconnection Networks, Rearrangeability, Routing Algorithms, Layered Cross Product.

1 Introduction

An N input multistage interconnection network (MIN) is called a *rearrangeable network* if it realizes every one of the $N!$ permutations in a single pass.

MINs consisting of $\log N$ stages (Omega, Flip, Indirect Binary Cube, Baseline, Butterfly) are all not rearrangeable. 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 is an example of this class of MINs and it is usually represented as the concatenation of a Baseline and a Reverse Baseline. One method of rearranging the switches of the Beneš network is known as the Looping algorithm [6]. It can, in general, be used for all symmetric MINs, as Beneš network. Nevertheless, for a a general Beneš equivalent MIN not having this symmetry property (e.g. the Omega-Omega^{-1}) the Looping algorithm is not applicable; moreover, for many $2\log N - 1$ stage MINs it is neither known whether they are rearrangeable or not. For the Omega-Omega^{-1}, Lee [4, 5] proposed a routing algorithm based on the decomposition of the first half of the network and the self-routing on the second half. Both this algorithm and the Looping algorithm have $O(N \log N)$ time complexity. The main problem of these two schemes is their non portability, that is they work only on the networks for which they are designed since they provide a control scheme implementing the routing that exploits the network configuration.

Yeh and Feng [7] proposed a routing scheme realizing any arbitrary permutation on a class of $2\log N$ stage MINs. Networks in this class are obtained by connecting two $\log N$ stage MINs by means of a stage that can assume various connection patterns in order to make the network symmetric. The routing scheme requires $O(N \log N)$ time. In [3] Feng and Seo consider the $2\log N$ stage concatenation of Omega-Omega and show that it can be converted in a Omega-Omega^{-1} or a Omega^{-1}-Omega by modifying the center stage. The time complexity is $O(N)$ but the algorithm can be applied only on symmetric $2\log N$ stage Omega-based networks.

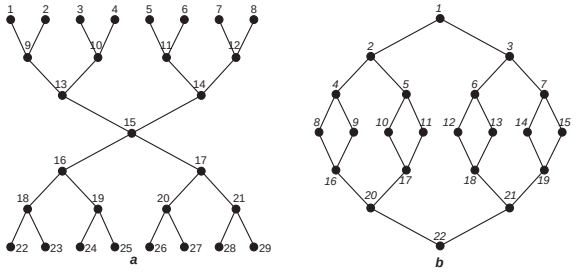

Figure 1: a. A $\begin{smallmatrix}\nabla\\ \Delta\end{smallmatrix}$ and b. a $\begin{smallmatrix}\Delta\\ \nabla\end{smallmatrix}$, both of dimension 8.

In this work we approach the problem of rearrangeability from a novel point of view, taking into account the partition into equivalence classes of the set of $2\log N - 1$ stage MINs [1]. This partition is developed from the concept of Layered Cross Product (LCP) [2], defined as follows:

An *l-layered graph*, $G = (V_1, V_2, \ldots, V_l, E)$ consists

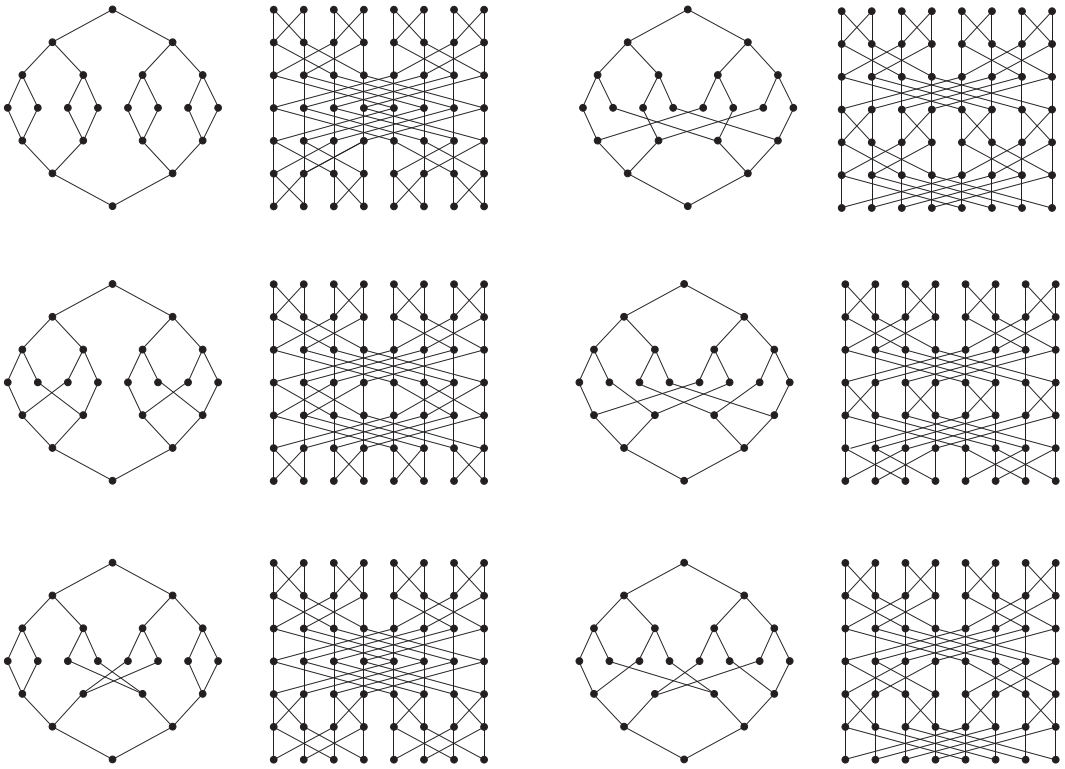

Figure 2: All the possible $\hat{\triangle}$s related to $N = 16$ and a representative of the corresponding equivalence class.

of l layers of nodes; V_i is the (non-empty) set of nodes in layer i, where $1 \leq i \leq l$; E is a set of edges: every edge connects nodes of two consecutive layers. Given two l-layered graphs $G' = (V_1', V_2', \ldots, V_l', E')$ and $G'' = (V_1'', V_2'', \ldots, V_l'', E'')$, their *Layered Cross Product* (LCP for short) , $G' \otimes G''$ is an l-layered graph $G = (V_1, V_2, \ldots, V_l, E)$ where V_i is the cartesian product of V_i' and V_i'', $1 \leq i \leq l$, 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 *second factor* of G, respectively.

We call *decomposition in factors* the inverse operation of the LCP. Each MIN can be decomposed into two factors (for an example, see Fig. 1), whose the first one, call it $\hat{\nabla}$, is always the same and it is represented by two complete binary trees sharing their root; the other factor, $\hat{\triangle}$, characterizes each class; all possible cases for $N = 16$ are shown in Fig. 2, together with a representative MIN of the corresponding class. We are able to extract those structural properties useful for the rearrangeability from the $\hat{\triangle}$ factor only. To focus our attention on a whole class instead of on a single MIN allows us to design more general algorithms.

2 A New Routing Algorithm for Beneš Equivalent MINs

In this section we propose a new algorithm for setting the switches of a Beneš equivalent MIN in order to satisfy any permutation Π. Note that the proposed algorithm runs on each MIN belonging to the class of Beneš equivalent MINs, apart from its symmetry. Therefore, the Looping algorithm is a particular case of this more general algorithm.

Before going on, we need to define some concepts. Given a MIN G, a *request for* G is an ordered pair (u, v), $u, v \in \{1, \ldots, N\}$, comprising an input u and an output v.

A *permutation* Π *for* G is a set of N requests for G having neither inputs nor outputs in common.

G *satisfies* Π if there exists a set of N edge disjoint paths from the input to the output of each request in Π passing through exactly one node in each stage.

A *legitimate path* from u to v is a path on G starting from input u, passing through exactly one node in each stage, and ending in output v.

Observe that the inputs and outputs of MINs are not involved in the definition of LCP, but it is not restrictive to add N inputs and N outputs both to the MINs and to their factors at the end of the computation of the LCP (see Fig. 3). This addition, although

not necessary, is very useful to highlight the start and the end of each legitimate path, both on the MIN and on its factors.

Exploiting the LCP, we are able to handle the rearrangeability of a MIN working only on its factors. To this aim, we state the following characterization, whose proof is omitted in this extended abstract.

Lemma 1 *A MIN $G = G_1 \otimes G_2$ is rearrangeable if and only if there is no permutation $\Pi = \{[1,\pi(1)],[2,\pi(2)],\ldots,[N,\pi(N)]\}$ such that two among its requests, let they be $[i,\pi(i)]$ and $[j,\pi(j)]$, correspond to two legitimate paths $P(i,\pi(i))$ and $P(j,\pi(j))$ whose pairs of factors $P_1(i,\pi(i))$ and $P_1(j,\pi(j))$, $P_2(i,\pi(i))$ and $P_2(j,\pi(j))$ both share an edge at the same edge stage.*

We have to take into account this lemma in order to choose, among all the possible legitimate paths, those leading to a switch setting satisfying Π. In other words this is equivalent to construct a new permutation Π' such that the "upper" Baseline equivalent MIN satisfies Π' top-down and the "lower" Baseline equivalent satisfies Π' bottom-up. In particular, given a Beneš equivalent MIN and a permutation Π, our algorithm builds legitimate paths stage by stage for each request of Π such that legitimate paths are pairwise edge disjoint. To do that, it builds the factors of each legitimate path on the factors of the MIN $\overset{\Delta}{\nabla} \otimes \overset{\nabla}{\Delta}$. In fact, $\overset{\nabla}{\Delta}$ is a tree and therefore all its legitimate paths are unique. It follows that we can impose conditions on $\overset{\Delta}{\nabla}$ only. Hence, the route followed by $P(i,\pi(i))$ uniquely depends on its second factor $P_2(i,\pi(i))$.

In order to avoid that the legitimate paths of each pair of requests $[i,\pi(i)]$ and $[j,\pi(j)]$, share an edge at the first edge stage, when we construct the factors of the legitimate path, stage by stage, we must impose that

a) for $i = 2k-1$ and $j = 2k$ (or vice-versa) for some $k \in 1,\ldots,\frac{N}{2}$, $P_2(i,\pi(i))$ goes down towards the left hand and $P_2(j,\pi(j))$ goes down towards the right hand (or vice-versa).

Analogously, in order to avoid that $P_2(i,\pi(i))$ and $P_2(j,\pi(j))$ share and edge at the last edge stage, we must impose that:

b) for $\pi(i) = 2k-1$ and $\pi(j) = 2k$ (or vice-versa) for some $k \in 1,\ldots,\frac{N}{2}$, $P_2(i,\pi(i))$ goes up towards the left hand and $P_2(j,\pi(j))$ goes up towards the right hand (or vice-versa).

Analogous conditions must hold in correspondence of each edge stage of the MIN.

An algorithm taking into account these considerations is the following. We explain it both formally and by an example. We choose to run our algorithm on a Omega-Omega^{-1} network, in order to highlight that it works on it, differently from the Looping algorithm, that correctly computes a switch setting for symmetric and recursive MINs only.

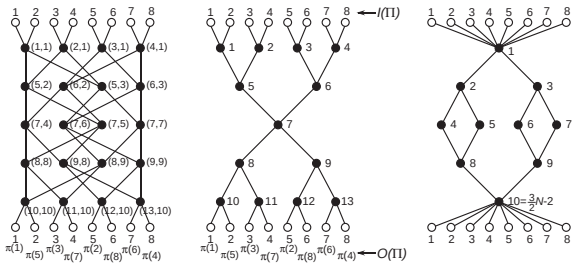

Figure 3: An example: a MIN with 8 inputs obtained concatenating an Omega and its reverse and a permutation Π to be satisfied.

Algorithm Routing on $\overset{\nabla}{\Delta} \otimes \overset{\Delta}{\nabla}$

Input: a MIN $G = \overset{\nabla}{\Delta} \otimes \overset{\Delta}{\nabla}$ and a permutation Π;
Output: a setting for all switches of G, s.t. G satisfies Π;

1. Construction of $I(\Pi)$ and $O(\Pi)$
(See Fig. 3.a and 3.b) Consider each node $(i,1)$ of the first stage of G; let x_i and x_i+1 be its inputs; then assign to the corresponding node i of the first layer of $\overset{\nabla}{\Delta}$ the same inputs; let $I(\Pi)$ be the obtained sequence of the inputs of $\overset{\nabla}{\Delta}$. Analogously, consider each node $(i,\frac{3}{2}N-2)$ of the last stage of G; let y_i and y_i+1 be its outputs; then assign to the corresponding node i of the last layer of $\overset{\nabla}{\Delta}$ the same outputs; let $O(\Pi)$ be the obtained sequence of the outputs of $\overset{\nabla}{\Delta}$. Observe that, even if in Fig. 3 $I(\Pi)$ coincides with the ordered sequence, this is – in general – not true.

For sake of clearness, from now on, we will highlight which instructions are performed on the MIN and which ones on the pair of sequences $(I(\Pi),O(\Pi))$.

2. Coupling elements in $I(\Pi)$ and $O(\Pi)$
(On $(I(\Pi),O(\Pi))$, see Fig. 4.a) Couple the $(2k-1)$-th and the $2k$-th element of $I(\Pi)$, $1 \le k \le \frac{N}{2}$. In the same way, couple the $(2k-1)$-th and the $2k$-th element of $O(\Pi)$, $1 \le k \le \frac{N}{2}$. From now on, given a $j \in I(\Pi)$, we define $M_I(j)$ its mate, i.e. the element belonging to the couple which j belongs to. In the example of Fig. 4.a, $M_I(6) = 5$, $M_I(3) = 4$ and so on. Analogously, given a $\pi(j) \in O(\Pi)$, we define $M_O(\pi(j))$. Some examples are: $M_O(\pi(5)) = \pi(1)$ and $M_O(\pi(1)) = \pi(5)$.

3. Labeling $I(\Pi)$ and $O(\Pi)$
(On $(I(\Pi),O(\Pi))$, see Fig. 4.a and 4.b) Let *Layer* be the currently considered stage of switches in G; at the beginning *Layer*$= 1$;

let $j \in I(\Pi)$; at the beginning $j = 1$;
repeat until all inputs and outputs are labeled
    if j is not labeled yet, then

label $j \in I(\Pi)$ with 'Left';
label $\pi(j) \in O(\Pi)$ with 'Left';
label $M_O(\pi(j)) = \pi(j')$ with 'Right';
label $j' \in I(\Pi)$ with 'Right';
$j \leftarrow M_I(j') \in I(\Pi)$;
else choose a non-labeled input as new j;
end repeat (see Fig. 4.c).

4. Switch setting of the current stage
(On the MIN, see Fig. 5.a) Set each of the $\frac{N}{2}$ switches at stage *Layer* to straight if its left and right inputs are labeled 'Left' and 'Right' in $I(\Pi)$, respectively; set the switch to cross otherwise; let the sequence $I(\Pi)$ pass through switches of stage *Layer*, so that an opportune permutation of $I(\Pi)$ becomes the input sequence of stage *Layer*+1.

5. Updating
(On $(I(\Pi), O(\Pi))$, see Fig. 4.d) Ordinately, consider from left to right each $j \in I(\Pi)$ labeled 'Left' and let the resulting sequence be $I_L(\Pi)$. Analogously, ordinately, consider from left to right each $j \in I(\Pi)$ labeled 'Right' and let the resulting sequence be $I_R(\Pi)$. Do the same for $O(\Pi)$ building $O_L(\Pi)$ and $O_R(\Pi)$;
$I(\Pi) \leftarrow I_L(\Pi)$ concatenated with $I_R(\Pi)$;
$O(\Pi) \leftarrow O_L(\Pi)$ concatenated with $O_R(\Pi)$;
Layer $\leftarrow$ *Layer*+1.
Observe that the new sequence $I(\Pi)$ is constituted by a set of pairs corresponding to the pairs of inputs of nodes at stage *Layer* (cf. Figs. 4.d and 5.a).

6. Iteration
(See Fig. 5.b) Repeat all steps from 2 to 5 until *Layer*$= \log N$.

7. Switch setting of the last $\log N$ stages
(See Fig. 5.c) All switches at stages $1, \ldots, \log N - 1$ of G are set; the rest of the MIN is a Banyan network having as input sequence an opportune permutation of $I(\Pi)$ and therefore all the remaining switches can be univocally set.

Theorem 2 *Given a MIN* $G = \frac{\nabla}{\Delta} \otimes \frac{\triangle}{\nabla}$ *and any permutation* Π, $O(N \log N)$ *time is sufficient for setting its switches s.t.* G *satisfies* Π.

Proof We prove the statement by proving the correctness of the algorithm **Routing on** $\frac{\nabla}{\Delta} \otimes \frac{\triangle}{\nabla}$ and, consequently, the rearrangeability of the class of MINs, and by studying its complexity.

Let $n = \log N$. The proof is by induction on n. If $n = 1$ the MIN consists of a single node and the result is obvious. Hence assume that the result is true for a MIN with $N/2$ inputs and prove for a MIN with N inputs. The key idea is to observe that the middle $2n - 3$ layers of $\frac{\triangle}{\nabla}$ (i.e. eliminating the two roots) are two smaller $\frac{\triangle}{\nabla}$s. Hence, it will be sufficient to decide whether each second factor of the legitimate path is to be routed through the left or the right sub-$\frac{\triangle}{\nabla}$. The

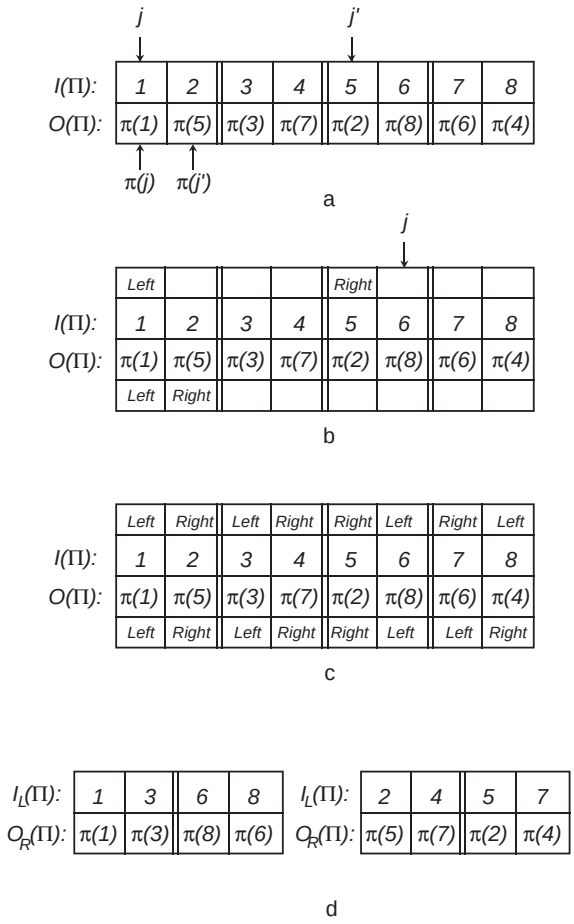

Figure 4: An example: a. $I(\Pi)$ and $O(\Pi)$; b. first step of their labeling; c. completion of their labeling; d. division into $I_L(\Pi)$ and $I_R(\Pi)$, $O_L(\Pi)$ and $O_R(\Pi)$.

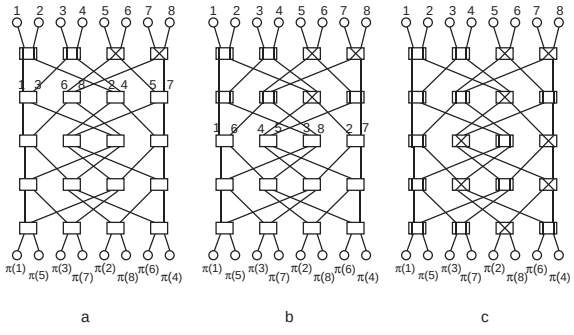

Figure 5: An example: a. setting the switches of the first stage; b. setting the switches of stages $1 \ldots \log N - 1$; c. setting all the switches of the MIN.

only constraints that we must satisfy to choose the left or the right sub-$\frac{\Delta}{\nabla}$, both upward and downward, are conditions a) and b). In other words, if the first factors of any two legitimate paths share a node at the first layer then their second factors must go to different sub-$\frac{\Delta}{\nabla}$s, and if the first factors of any two legitimate paths share a node at the last layer then their second factors must come from different sub-$\frac{\Delta}{\nabla}$s. These constraints are satisfied by step 3 of the algorithm. It remains to prove that we can always assign each second factor of the legitimate path to the left or right sub-$\frac{\Delta}{\nabla}$ in a way that satisfies the constraints a) and b) and that this assignment is feasible, i.e. the consequent switch setting routes Π. The proof that a) and b) are satisfied derives from two facts:

i) the set of elements in $I(\Pi)$ and $O(\Pi)$ considered during step 3 induces even cycles;

ii) for each $j \in I(\Pi)(O(\Pi))$ routed towards a sub-$\frac{\Delta}{\nabla}$, its mate is routed towards the other sub-$\frac{\Delta}{\nabla}$.

The proof that the assignment is feasible derives from two facts:

i) for each $j \in I(\Pi)$ routed towards a sub-$\frac{\Delta}{\nabla}$, $\pi(j) \in O(\Pi)$ is routed towards the same sub-$\frac{\Delta}{\nabla}$;

ii) for each $\pi(j') \in O(\Pi)$ routed towards a sub-$\frac{\Delta}{\nabla}$, $j' \in I(\Pi)$ is routed towards the same sub-$\frac{\Delta}{\nabla}$.

Each sub-$\frac{\Delta}{\nabla}$ can be handled by the inductive hypothesis and therefore the correctness is proved.

About the time complexity, we divide the analysis step by step. Step 1 runs in $O(N)$ time. Steps 2, 3, 4 and 5 all run in $O(N)$ time and they are repeated $O(\log N)$ times by Step 6. For what concerns Step 7, the requests are self-routed and then $O(N \log N)$ time is sufficient. Hence, the global time complexity is $O(N \log N)$. **Q.E.D.**

Remark The previous theorem provides a constructive rearrangeability proof for the whole class of MINs decomposable as $\frac{\nabla}{\Delta} \otimes \frac{\Delta}{\nabla}$; inside this class we highlight the Beneš network, the Double Baseline and all those obtained by concatenating a Baseline equivalent and its reverse (e.g. Baseline-Reverse Baseline, Omega-Omega^{-1}, Butterfly-Reverse Butterfly, etc.).

3 Rearrangeability of Another Class of MINs

Many interesting networks (e.g. the Double Butterfly, the Double Omega, etc.) fall outside the class obtained as LCP of $\frac{\nabla}{\Delta} \otimes \frac{\Delta}{\nabla}$. The decomposition in factors of all these networks has been studied in [1], and it has been proved that they can be decomposed as LCP of $\frac{\nabla}{\Delta}$ and a modified $\frac{\Delta}{\nabla}$. Namely, the merging of the leaves of Δ and of ∇ is not performed ordinately, but creating some crossings (see Fig. 6.c). For each N, the number of non equivalent modified $\frac{\Delta}{\nabla}$s that are a factor of a MIN is $(\log N - 1)!$ and each different modified $\frac{\Delta}{\nabla}$ leads to a different equivalence class of MINs [1].

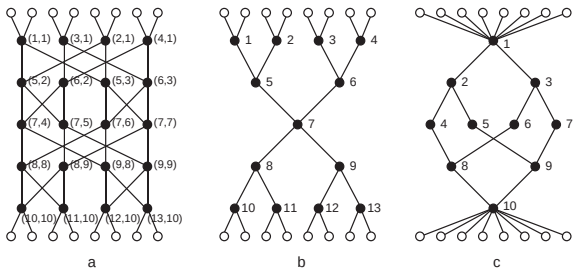

Figure 6: a. A MIN with 8 inputs constituted by two concatenated copies of Butterfly network; b. its first factor, i.e. a $\frac{\nabla}{\Delta}$; c. its second factor, i.e. a modified $\frac{\Delta}{\nabla}$.

Unfortunately, the algorithm described in the previous section cannot run on each equivalence class in view of the asymmetry of the modified $\frac{\Delta}{\nabla}$ with respect to the middle layer. Nevertheless, we can slightly modify the algorithm so that it can run on another class of MINs, proving also the rearrangeability of each network belonging to this class. A representative MIN belonging to this equivalence class of networks is the network obtained by concatenating an N input Reverse Butterfly and a Butterfly whose the first and second stages are reversed (see Fig. 6.a).

In this extended abstract we have no enough room to describe the modifications we have introduced to the algorithm.

4 Conclusions and Open Problems

In this paper we have provided an algorithm to realize any permutation Π by setting the switches of all MINs equivalent to the Beneš network, i.e. all networks decomposable as $\frac{\nabla}{\Delta} \otimes \frac{\Delta}{\nabla}$. With this algorithm we have given a constructive proof of rearrangeability for a wide class of networks, substituting all the different proofs presented in literature. The time complexity is the same as the Looping algorithm, i.e. $O(N \log N)$, but the main difference is that the Looping Algorithm is immediately applicable only on the Beneš Networks and on other networks built as two smaller copies of the same network plus two additive stages connecting them. Then we have proven the rearrangeability of a class of MINs whose rearrangeability was not known.

The idea under both algorithms is to perform a sort of routing with certain properties on the $\frac{A}{\nabla}$ factor only. Then, we translate this special routing performed on $\frac{A}{\nabla}$ in the switch setting of the considered MIN and, consequently, in the routing of the permutation.

The interest of this result lies in that the approach to rearrangeability, based on LCP, is completely new and it allows one to deal with a simplified structure (the $\frac{A}{\nabla}$ factor) instead of the whole MIN.

Although we extend the set of MINs known to be rearrangeable, for other classes of MINs (e.g. Omega-Omega equivalent) it is still not known whether they are rearrangeable. In fact, our approach requires to design an "ad hoc" routing algorithm for each equivalence class (i.e. for each different $\frac{A}{\nabla}$ factor), so an interesting future work is to develop an algorithm running on the $\frac{A}{\nabla}$ factor of the Omega-Omega.

References

[1] T. Calamoneri and A. Massini, Efficiently Checking the Equivalence of Multistage Interconnection Networks, *Proc. Parallel and Distributed Computing and Systems (PDCS'99)*, 1999, 23-30.

[2] S. Even and A. Litman, Layered Cross Product - A technique to construct interconnection networks, 4^{th} *ACM SPAA*, 1992, 60-69.

[3] T. Feng and S–W. Seo, A New Routing Algorithm for a Class of Rearrangeable Networks, *IEEE Trans. Comput.*, 43, 1994, 1270-1280.

[4] K. Y. Lee, On the Rearrangeability of $2 \log N - 1$-Stage Permutation Networks, *IEEE Trans. Comput.*, C34, 1985, 412-425.

[5] K. Y. Lee, A New Beneš Networks Control Algorithm, *IEEE Trans. Comput.*, C36, 1987, 768-772.

[6] B. C. Opferman and N. T. Tsao-Wu, On a Class of Rearrangeable Switching Networks, *Bell Syst. Tech. J.*, 50(5), 1971.

[7] Y. Yeh and T. Feng, On a Class of Rearrangeable Networks, *IEEE Trans. Comput.*, C41, 1992, 1361-1379.