Nordic Journal of Computing 1(1994), 202–213.

A TABULAR METHOD FOR VERIFICATION OF DATA EXCHANGE ALGORITHMS ON NETWORKS OF PARALLEL PROCESSORS

K. COOLSAET
Department of Pure Mathematics
University of Gent
Galglaan 2, B–9000 Gent
Belgium
kc@cage.rug.ac.be

V. FACK
Department of Applied Mathematics
and Computer Science
University of Gent
Krijgslaan 281–S9, B–9000 Gent
Belgium
Veerle.Fack@rug.ac.be

H. DE MEYER*
Department of Applied Mathematics
and Computer Science
University of Gent
Krijgslaan 281–S9, B–9000 Gent
Belgium

Abstract. A tabular method for verification of data exchange algorithms on networks which possess a certain symmetry (whose underlying graph is a Cayley graph) is given. The algorithms use no intermediate buffering of messages.

To illustrate this method, optimal total exchange (i.e., all-to-all personalised) algorithms are given for several much-used processor configurations, such as ring networks, the hypercube and symmetric meshes with wrap-around (two and three-dimensional). To the best of our knowledge the latter are new.

ACM CCS Categories and Subject Descriptors: F.2.2, G.2.2

Key words: total exchange, optimal algorithm, hypercube, symmetric meshes with wrap-around

1. Introduction

This section introduces some terminology. Most concepts defined here have been described in more detail in [8].

Consider a network of parallel processors. Such a network can be treated as an abstract *graph* with processors as *nodes* (vertices) and duplex communication channels as *links* (edges). Each connection between two adjacent

*Research Director at the National Fund for Scientific Research (N.F.W.O. Belgium)

Received November 1993. Accepted August 1994.

processors is treated as consisting of 2 different links, one link for each direction. Denote the set of nodes of a given network by N and the set of links by L.

A sequence of nodes $i_0, i_1, \ldots, i_k \in N$ where i_j and i_{j+1} are neighbours $(0 \leq j < k)$ is a *path* of length k between nodes i_0 and i_k. The *distance* between two nodes is the length of the shortest path that connects these nodes.

A *data exchange algorithm* is a set of programs (one for each processor in the network) which are executed in parallel and whose purpose it is to send data (in the form of messages) between several processors. A *total exchange algorithm* is a data exchange algorithm which allows every node i in the network to send a different message $m_{i,j}$ to every other processor j in the network [5, 11].

We concentrate on the order and the paths by which the messages are sent across the network, and not on the lower level processing needed to make these transfers. The abstract model we use, makes the following assumptions about timing : the transfer of a message from a node to a neighbouring node occurs only at discrete time intervals; the time taken for extra processing of messages within each node is negligible; and one node is allowed to transfer messages to all of its neighbouring nodes simultaneously, but at most one message at the same time across a given link, i.e., one message for each direction. (In practice this 'simultaneous' transmission of messages is allowed to take a small amount of time, as long as subsequent time intervals do not overlap. Some sort of extra synchronisation might also be necessary. Other authors use a similar approach [3].)

By definition, an algorithm does not need *intermediate buffering* of messages if every message $m_{i,j}$ which at a given time T arrives at a node k $(\neq j)$, leaves that node again at time $T + 1$.

The following definitions are borrowed from graph theory [6, 10]. A 1-to-1 mapping $\sigma : N \to N$ is an *automorphism* of the network if two nodes $\sigma(i)$ and $\sigma(j)$ are linked if and only if the nodes i and j are linked. The set of automorphisms of a network forms a group. A subgroup G of the automorphism group *acts regularly* on the graph, if for every pair i, j of nodes there exists exactly one element σ of G such that $\sigma(i) = j$. We call a network for which such a group G exists, a *Cayley* network (after similarly named Cayley graphs [6]). It is not always easy to determine whether a network is a Cayley network and to find an appropriate group G. However, most networks considered in the literature (e.g., hypercubes, meshes with wrap-around, cyclic networks, etc.) are Cayley networks with obvious groups (cf. section 3). For that reason Cayley graphs have been used for quite some time in the analysis of interconnection networks [1, 2, 3, 4, 7].

In a Cayley network we may take the group G itself as the nodeset N. If $S = \{g_1, \ldots, g_d\}$ is the set of neighbours of the identity $1 \in G = N$, then $g, h \in G$ are neighbours if and only if $g^{-1}h \in S$. Every Cayley network is uniquely determined by its group G and the set S.

For each path $h_0, \ldots, h_n$ in a Cayley network there is a corresponding *word* $w = g_1 \cdots g_n \in S^*$ (i.e., a word with elements in S), where $g_i \stackrel{\text{def}}{=} h_{i-1}^{-1} h_i \in S$. We have $h_0 w = h_n$. Conversely, every word $w \in S^*$ determines a path joining a node h_0 with the node $h_0 w$.

A data exchange algorithm on a Cayley network (G, S) is *locally defined* iff for every $g \in G$ and for every message $m_{h,k}$ $(h, k \in G)$ which is transferred by the algorithm from node 1 to its neighbour $g_i \in S$ at a given time T, a message $m_{gh,gk}$ is transferred at the same time T from node g to its neighbour gg_i. Note that the word w that corresponds to the path taken by $m_{h,k}$ is the same as that corresponding to the path taken by $m_{gh,gk}$. Hence, in order to study locally defined algorithms, it is only necessary to investigate what happens at a *single node* (usually the node corresponding to the identity $1 \in G$).

The following lemma, which we proved in [8], provides us with an important tool :

LEMMA 1. *Consider a locally defined algorithm on a Cayley network (G, S). Let m be a message with associated word $w = g_i w' \in S^*$ sent from a node $h \in G$ at a given time. Then in the same time interval a message m' arrives at h from the node hg_i^{-1}. The path which m' still has to travel to reach its destination corresponds to the word w'. (If w' is empty, then h is the destination of the message m'.)*

2. Verification of data exchange algorithms without intermediate buffering

The above lemma can be used to investigate the traffic of messages during a data exchange algorithm. Indeed, let $w_1, \ldots, w_k$ be the words corresponding to messages that are still waiting to be sent from node 1 at time T. Assume that the messages sent at time T correspond to the words $w_1, \ldots, w_j$ $(j \leq k)$. (Note that the first element of each of these j words must be different, for this indicates the link across which the corresponding message needs to be sent.) By lemma 1, at time $T + 1$ the messages that are waiting at node 1 to be sent now correspond to the words $w'_1, \ldots, w'_j, w_{j+1}, \ldots, w_k$, where all empty words have to be removed. (We use the w'-notation of the lemma.)

If the data exchange algorithm does not use intermediate buffering, then the messages sent at time $T + 1$ must include those just received, i.e., the messages that correspond to $w'_1, \ldots, w'_j$. This is only possible when all first elements of these words differ. The following theorem provides us with an easy way to verify this :

THEOREM 1. *Every locally defined data exchange algorithm without intermediate buffering on a Cayley network (G, S) corresponds to a rectangular table with $\#S$ rows whose entries are either blank or elements of S with the following properties :*

(1) Every row consists of zero or more words in S^* separated by zero or more blanks.

(2) Every column contains each element of S at most once.

The messages sent by this algorithm are exactly those that correspond to the words in the table. The total time taken by the algorithm is equal to the number of columns in the table.

Conversely, every such table corresponds to a locally defined data exchange algorithm without intermediate buffering in the following way : Remove the first $T-1$ columns of the table. The words that begin in the first column of the table thus obtained correspond to the messages which are transferred from a given node at time T. Each message is sent to the neighbour that is associated with the first element of the corresponding word.

The proof of this theorem is an immediate consequence of lemma 1. We leave the details to the reader. This technique is an extension of a similar technique used in [8].

3. Applications

We apply the above technique to several much-used processor configurations. Some of the algorithms in section 3.1 and 3.2 are also given in [8] (be it in a different format).

To our knowledge, the algorithms given in sections 3.3 and 3.4 are new algorithms for optimal total exchange without buffering on that kind of network. In [5] optimal total exchange algorithms for wrap-around meshes are given, but these use buffering.

3.1 Ring networks

A network of 6 processors connected in a ring can be considered as a Cayley network with the cyclic group C_6 :

$$G \sim C_6 = \langle a \mid a^6 = 1 \rangle$$

and with link set $S = \{a, \bar{a} \stackrel{\text{def}}{=} a^{-1}\}$. In figure 1 a representation of this network is given.

Consider a data exchange algorithm in which every node must send a message to each node at *odd distance*. The messages sent from a given node correspond to the words $a, \bar{a}$ and aaa (or $\bar{a}\bar{a}\bar{a}$). The following is a table that corresponds to such an algorithm :

a	a	a	a
$\bar{a}$			

This algorithm takes time 4. The blank spaces indicate that during the last 3 time intervals only one link per processor is used, instead of the available two.

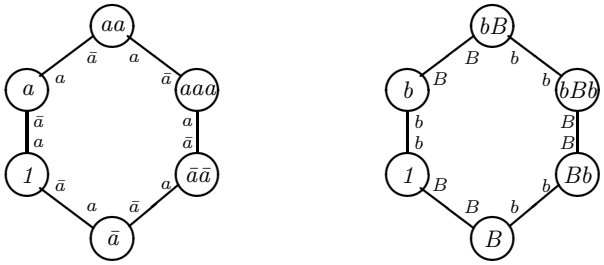

Fig. 1: *Representations of a hexagonal network using groups C_6 (left) and D_6 (right)*

On the same network, a *total* exchange algorithm (i.e., with messages $a, aa, aaa, \bar{a}, \bar{a}\bar{a}$) might be represented as follows :

a	a	a	a	a	a
$\bar{a}$	$\bar{a}$	$\bar{a}$			

This algorithm takes time 6. Again, during the last 3 time intervals only one link per processor is used instead of the available two.

The hexagonal network can also be represented using a different group, the diheder group D_6 :

$$G \sim D_6 = \langle b, B \mid b^2 = B^2 = (bB)^3 = 1 \rangle$$

and link set $S = \{b, B\}$. In figure 1 also this representation of the network is given.

Note that the group D_6 is not commutative. Hence, the words bB and Bb apply to different nodes. However bBb and BbB designate the same node, although they determine a different path to that node.

Using this representation a faster 'odd distance' data exchange algorithm can be found, needing only 3 units of time :

b	B	b
B	b	

Similarly, a faster total exchange algorithm, needing only 5 units of time, is provided by the following table :

b	B	b	b	B
B	b	B		b

In general, given a ring network of n processors, an optimal total exchange algorithm without buffering can always be found.

If n is odd, we use the cyclic group $C_n = \langle x \mid x^n = 1\rangle$ and link set $S = \{x, \bar{x} \stackrel{\text{def}}{=} x^{-1}\}$. The corresponding algorithm table is the following :

x	x	x	...	x	x	$\cdots$	x
$\bar{x}$	$\bar{x}$	$\bar{x}$	...	$\bar{x}$	$\bar{x}$	$\cdots$	$\bar{x}$

The total time taken by the algorithm is $\frac{1}{8}(n-1)(n+1)$.

If n is even, we use the diheder group $D_n = \langle y, Y \mid y^2 = Y^2 = (yY)^{n/2} = 1\rangle$ with link set $S = \{y, Y\}$. We must distinguish further between $n/2$ even or odd. Below we give the table for $n = 12$:

y	Y	y	Y	y	Y	y	Y	Y	y	Y	y
Y	y	Y	y	Y	y	Y	y	y	Y	y	Y

Y	y	Y	y	Y	y	Y	y	y	Y	y	Y
y	Y	y	Y	y	Y	y	Y	Y	y	Y	y

The table for $n = 10$ can be obtained from this table by deleting the rightmost two words :

y	Y	y	Y	y	Y	y	Y	Y	y	Y	y	
Y	y	Y	y	Y	y	Y	y	y	Y	y	Y	y

The general case can easily be derived from these examples. The total time taken is $\frac{1}{8}n^2$ when $n/2$ is even, and $\frac{1}{8}(n^2+4)$ otherwise.

3.2 The hypercube

We proved in [8] that an optimal total exchange algorithm without buffering always exists for the hypercube of any dimension. Below we give the example for dimension 4, i.e., a hypercubic network of 16 processors. This is a Cayley network with group

$$\begin{aligned} G \sim C_2^4 \;=\; & \langle a, b, c, d \mid a^2 = b^2 = c^2 = d^2 = 1, \\ & ab = ba, ac = ca, ad = da, bc = cb, bd = db, cd = dc\rangle \end{aligned}$$

and link set $S = \{a, b, c, d\}$. In the usual binary notation used for hypercubes, the elements of S correspond to 1000, 0100, 0010 and 0001. Group multiplication then corresponds to bitwise exclusive or.

We give two examples of total exchange algorithms for this network :

a	d	a	c	a	b	c	d	a	b	c	d	b	c	b	d
b	a	b	d	b	c	d	a	b	c	d	a	c	d	c	a
c	b	c	a	c	d	a	b	c	d	a	b	d	a	d	b
d	c	d	b	d	a	b	c	d	a	b	c	a	b	a	c

The tables contain no blanks and use the shortest possible message paths, which proves that the algorithms are optimal. In general, for an n-dimensional hypercube, the total time taken is 2^{n-1}.

3.3 Two-dimensional symmetric meshes with wrap-around

A network of n^2 processors connected in a symmetric mesh with wrap-around can be considered as a Cayley network with commutative group

$$G \sim C_n^2 = \langle a, b \mid a^n = b^n = 1, ab = ba \rangle$$

and link set $S = \{a, b, \bar{a} \stackrel{\text{def}}{=} a^{-1}, \bar{b} \stackrel{\text{def}}{=} b^{-1}\}$. When n is odd, this group can be used to obtain an optimal total exchange algorithm.

In a total exchange algorithm every node needs to send $n^2 - 1$ messages. Because of the rotational symmetry of the network, these messages can be partitioned into 4 equivalent sets, where each set is transformed into another set by the substitution

$$\sigma : a \to b \to \bar{a} \to \bar{b} \to a$$

This rotational symmetry provides us with an easy way to setup an algorithm table. Indeed, the topmost line of the table consists of all messages of the first set, while the other lines are obtained by applying σ to the line above. This yields a total exchange algorithm taking minimal time $\frac{1}{8}n(n-1)(n+1)$.

As an example we consider the network for $n = 5$, of which a representation is given in figure 2. The first set (framed in the figure) contains the messages

$$a, aa, ab, aab, abb, aabb.$$

(Because of commutativity, we might as well have chosen other message paths, e.g., a, aa, ba, aba, bab, $abab$. This is correct, as long as the other three sets are renamed accordingly.)

The corresponding algorithm table is the following :

a	a	a	a	b	a	a	b	a	b	b	a	a	b	b
b	b	b	b	$\bar{a}$	b	b	$\bar{a}$	b	$\bar{a}$	$\bar{a}$	b	b	$\bar{a}$	$\bar{a}$
$\bar{a}$	$\bar{a}$	$\bar{a}$	$\bar{a}$	$\bar{b}$	$\bar{a}$	$\bar{a}$	$\bar{b}$	$\bar{a}$	$\bar{b}$	$\bar{b}$	$\bar{a}$	$\bar{a}$	$\bar{b}$	$\bar{b}$
$\bar{b}$	$\bar{b}$	$\bar{b}$	$\bar{b}$	a	$\bar{b}$	$\bar{b}$	a	$\bar{b}$	a	a	$\bar{b}$	$\bar{b}$	a	a

In case n is even, the group C_n^2 cannot provide us with an optimal total exchange algorithm, for similar reasons as with the hexagonal network. However, the network can also be represented using a different group. For example :

$$G \sim D_n^2 \quad = \quad \langle a, b, A, B \mid a^2 = b^2 = A^2 = B^2 = (aA)^{n/2} = (bB)^{n/2} = 1, \\ ab = ba, aB = Ba, Ab = bA, AB = BA \rangle$$

with link set $S = \{a, b, A, B\}$. (Note that for $n > 4$ this group is not commutative, hence $aAbB$ is not the same as $AaBb$. However $aAbB = bBaA$!) A representation of the case $n = 6$ is shown in figure 3.

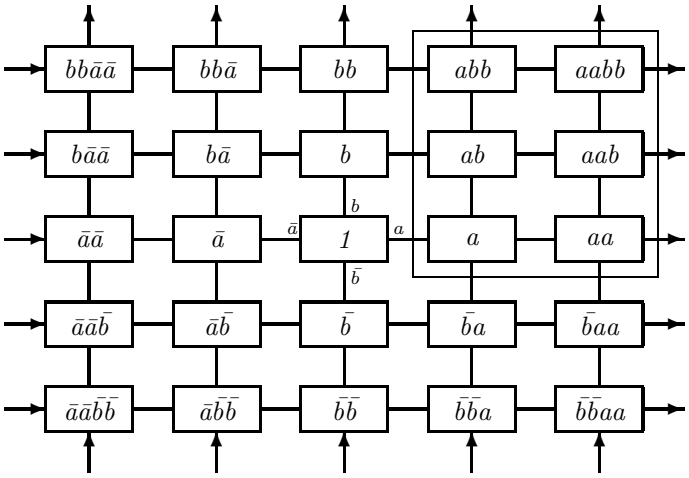

Fig. 2: *Representation of a symmetric mesh with wrap-around of odd order ($n = 5$)*

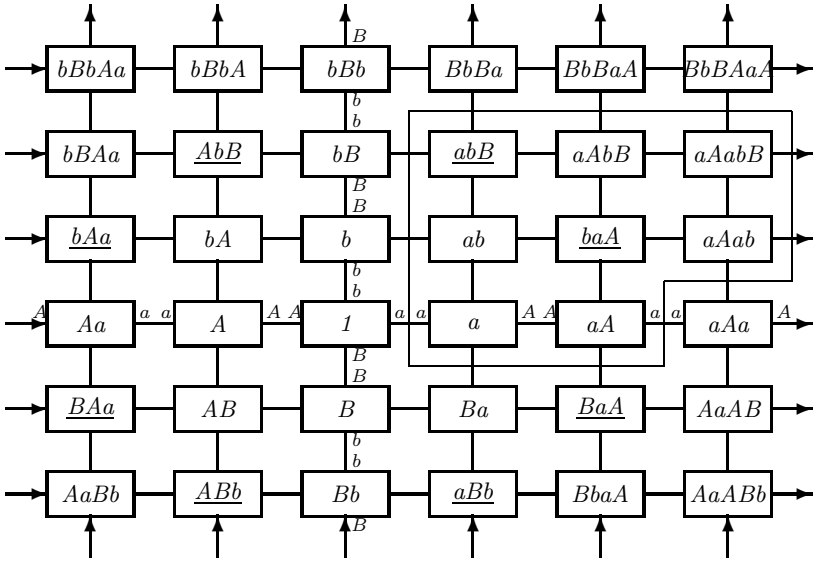

Fig. 3: *Representation of a symmetric mesh with wrap-around of even order ($n = 6$)*

Because $n^2 - 1$ is not divisible by 4, it is not possible to use σ as in the odd case. The rotational symmetry breaks down because of the 3 special messages $aAaA\cdots$, $bBbB\cdots$ (both of length $n/2$) and their product $aAaA\cdots bBbB\cdots$ (of length n). Indeed, we have

$$\begin{array}{rcll} \sigma(\sigma(aAaA\cdots)) & = & AaAa\cdots & = & aAaA\cdots \\ \sigma(\sigma(bBbB\cdots)) & = & BbBb\cdots & = & bBbB\cdots \\ \sigma(aAaA\cdots bBbB\cdots) & = & bBbB\cdots AaAa\cdots & = & aAaA\cdots bBbB\cdots \end{array}$$

Therefore, if we leave out these 3 messages, the corresponding data exchange algorithm could be built using σ. In figure 3 the messages of the first set are framed.

To construct a *total* exchange algorithm we set apart these 3 special messages together with 8 more. These 8 messages are those messages of length $n/2$ which contain exactly one of $\{a, A\}$ or exactly one of $\{b, B\}$. In figure 3 the relevant words are underlined.

Clearly this set of 8 words is rotationally symmetric. Hence, if we leave out all 11 messages, the remainder can still be put into a valid algorithm table which will constitute the first part of the table for the total exchange algorithm. The second part of the algorithm table contains the 11 messages (after some renaming), in the following way :

a	A	a	A	$\cdots$	b	B	b	B	$\cdots$	B	a	A	a	$\cdots$
b	a	A	a	$\cdots$	a	b	B	b	$\cdots$	b	A	a	A	$\cdots$
A	B	b	B	$\cdots$	B	A	a	A	$\cdots$	A	b	B	b	$\cdots$
B	b	B	b	$\cdots$	A	a	A	a	$\cdots$	a	B	b	B	$\cdots$

For $n \geq 6$ this approach yields a total exchange algorithm with minimal time $\frac{1}{8}n^3$. For $n = 4$ we cannot use the same method. However, this case corresponds to the hypercube of dimension 4 (indeed, $D_4 \sim C_2^2$) which has been treated in section 3.2.

A total exchange algorithm for the case $n = 6$ is given by the following table :

a	a	b	a	A	a	A	a	b	a	A	b	B	a	A	a	b	B	a	A	a	A	a	A
b	b	A	b	B	b	B	b	A	b	B	A	a	b	B	b	B	a	b	a	A	a	b	B	b	A	a
A	A	B	A	a	A	a	A	B	A	a	B	b	A	a	A	a	b	A	B	b	B	A	a	A	b	B
B	B	a	B	b	B	b	B	a	B	b	B	A	B	b	B	b	A	B	b	B	A	a	A	a	b	B

3.4 Three-dimensional symmetric meshes with wrap-around

As in the previous section we distinguish between n odd and n even. For n odd, the network may be represented by the commutative group

$$G \sim C_n^3 = \langle a, b, c \mid a^n = b^n = c^n = 1; a, b, c \text{ commute} \rangle$$

and link set $S = \{a, b, c, \bar{a} \stackrel{\text{def}}{=} a^{-1}, \bar{b} \stackrel{\text{def}}{=} b^{-1}, \bar{c} \stackrel{\text{def}}{=} c^{-1}\}$. The messages sent by a total exchange algorithm are of the form $a^i b^j c^k$, with $-n/2 < i, j, k < n/2$,

where we have used the shorthand notation a^i $(i > 0)$ for the word $a \cdots a$ of length i and the notation x^{-i} for $\bar{x}^i$.

We now use the substitution

$$\sigma : a \to b \to c \to \bar{a} \to \bar{b} \to \bar{c} \to a$$

This partitions the messages in classes most of which have size six. The exceptions are the messages of the form $a^i b^{-i} c^i$ with $-n/2 < i < n/2$, which come in classes of size two. Now consider the following table, containing the messages of one class of size 2 and three classes of size 6, with $i > 0$:

a^i	c^i	$\bar{b}^i$	b^i	$\bar{a}^i$	$\bar{c}^i$
$\bar{a}^i$	$\bar{b}^i$	a^i	c^i	a^i	b^i
$\bar{b}^i$	$\bar{c}^i$	$\bar{a}^i$	$\bar{c}^i$	b^i	c^i
b^i	$\bar{a}^i$	c^i	$\bar{a}^i$	$\bar{b}^i$	a^i
$\bar{c}^i$	b^i	$\bar{c}^i$	a^i	c^i	$\bar{b}^i$
c^i	a^i	b^i	$\bar{b}^i$	$\bar{c}^i$	$\bar{a}^i$

If we join all these tables for $0 < i < n/2$ and extend the result with the remaining classes of size 6, using σ in the manner of section 3.3, then this provides us with a total exchange algorithm with minimal time $\frac{1}{8}n^2(n-1)(n+1)$.

For $n > 4$ even, we use the group

$$\begin{aligned} G \sim D_n^3 \quad = \quad & \langle a, b, c, A, B, C \mid a^2 = b^2 = c^2 = A^2 = B^2 = C^2 = 1, \\ & (aA)^{n/2} = (bB)^{n/2} = (cC)^{n/2} = 1, \\ & \text{all generators commute, except } aA \neq Aa, bB \neq Bb, cC \neq Cc \rangle \end{aligned}$$

with link set $S = \{a, b, c, A, B, C\}$.

For ease of notation, denote by α_i the message $aAa \cdots$ of length i and by α_{-i} the message $AaAa \cdots$ of length i, and define $\beta_{\pm i}$ and $\gamma_{\pm i}$ in a similar way. The messages to be sent are now of the form $\alpha_i \beta_j \gamma_k$ with $-n/2 \leq i, j, k \leq n/2$. Note however that $\alpha_{n/2}$ and $\alpha_{-n/2}$ represent the same group element, and therefore when $|i| = n/2$, $|j| = n/2$ or $|k| = n/2$, we have to choose one of several shortest paths for the corresponding message.

Using the same σ as in the odd case, we may partition the messages into different classes. Most classes have size 6 with the following exceptions :

- The message $\alpha_{n/2}\beta_{n/2}\gamma_{n/2}$ forms a class of size 1.
- There are two classes of size 3 :

$$\{\alpha_{n/2}, \beta_{n/2}, \gamma_{n/2}\} \quad \text{and} \quad \{\alpha_{n/2}\beta_{n/2}, \beta_{n/2}\gamma_{n/2}, \gamma_{n/2}\alpha_{n/2}\}$$

- The $n-2$ messages of the form $\alpha_i \beta_{-i} \gamma_i$ $(-n/2 < i < n/2, i \neq 0)$ determine $n/2 - 1$ classes of size 2.

The classes of size 1 and size 3 can be placed into an algorithm table using two extra classes of size 6, i.e., the classes generated by $c\alpha_{n/2-1}$ and $B\alpha_{n/2-1}$, in the following way :

a	α_-	B	β_+	c	γ_-	b	α_-
A	α_+	b	β_-	C	γ_+	a	β_-
B	β_+	c	γ_-	a	α_-	C	β_+
C	γ_+	A	α_+	b	β_-	B	γ_+
c	β_-	a	γ_+	B	α_+	A	γ_-
b	γ_-	C	α_-	A	β_+	c	α_+

(We have used the shorter notation $\alpha_\pm$ for $\alpha_{\pm(n/2-1)}$. Note that $a\alpha_- = \alpha_{n/2}$, $A\alpha_+ = \alpha_{-n/2}$, etc.)

Now consider the classes of size 2. We first treat the cases $i = 1$ and $i = 2$. As $n > 4$, these cases always occur. We use two extra classes of size 6 to obtain the following table :

a	A	B	b	c	C	a	B	c
A	a	b	B	C	c	A	b	C
b	B	c	C	a	A	B	A	a
B	b	C	c	A	a	b	a	A
c	C	A	a	b	B	C	c	B
C	c	a	A	B	b	c	C	b

The extra classes are generated by the elements CaA and baA. Note that these extra classes are different from the ones used above, even for $n = 6$.

For $i \geq 3$ we need 2 extra classes of size 6 to obtain the following table (now using the notation $\alpha_\pm$ to denote $\alpha_{\pm(i-1)}$) :

a	α_-	B	β_+	c	γ_-
A	α_+	b	β_-	C	γ_+
C	β_+	A	γ_-	b	α_-
B	γ_+	c	α_+	a	β_-
c	β_-	a	γ_+	B	α_+
b	γ_-	C	α_-	A	β_+

In this case, the two extra classes are generated by $c\alpha_{i-1}$ and $B\alpha_{i-1}$. Because $3 \leq i < n/2$ there is no conflict with the classes used for the previous tables.

The complete total exchange algorithm is constructed by concatenating all tables displayed above, and adding the other classes of size 6 using σ as in the odd case. This results in an optimal algorithm taking time $\frac{1}{8}n^4$.

It easily seen that the case $n = 4$, which is not covered by the above, corresponds to the 6-dimensional hypercube, discussed in section 3.2.

4. Conclusion

The tabular method provides a useful tool for *verifying* given data exchange algorithms, but does however not provide a systematic method for *constructing* such algorithms, and the algorithms given above may therefore seem a little *ad hoc*.

Still, it provides an important simplification of the construction process. When searching for a data exchange algorithm on a given graph, one first has to choose a Cayley group (of which there might be several), next select a set of message words and, last but not least, fit these words into an algorithm table. The latter might be done by means of a computer program, but we have experienced that, to avoid combinatorial explosion, supplementary properties of the problem need to be taken into account, such as additional symmetry of the group.

We have used a similar method for the construction of data exchange algorithms for star graphs [9].

References

[1] S.B. Akers, D. Harel and B. Krishnamurthy, "The star graph : an attractive alternative to the n-cube", *Proc. International Conference on Parallel Processing*, 1987, 393–400.

[2] S.B. Akers and B. Krishnamurthy, "A group-theoretic model for symmetric interconnection networks", *IEEE Trans. Comput.* **38**(4) (1989), 555-566.

[3] F. Annexstein and M. Baumslag, "A unified approach to off-line permutation routing on parallel networks", *Proc. 2nd Annual ACM Symposium on Parallel Algorithms and Architectures*, 1990, 398–406.

[4] M. Baumslag, "Fault-tolerance properties of deBruijn and shuffle-exchange networks", *Proc. of the fifth IEEE Symposium on Parallel and Distributed Processing*, 1993, 556–563.

[5] D. Bertsekas and J. Tsitsiklis, *Parallel and distributed computation, Numerical methods*, Prentice Hall International Editions, 1989.

[6] N. Biggs, *Algebraic graph theory*, Cambridge Tracts in Mathematics **67**, Cambridge University Press, 1974.

[7] G.E. Carlsson, J.E. Cruthirds, H.B. Sexton and C.G. Wright, "Interconnection networks based on a generalization of the cube-connected cycles", *IEEE Trans. Comput.*, **C-34**(8) (1985), 769–772.

[8] K. Coolsaet, H. De Meyer and V. Fack, "Optimal algorithms for total exchange without buffering on the hypercube", *BIT* **32** (1992), 559–569.

[9] K. Coolsaet and V. Fack, "Optimal data exchange algorithms on star graphs", *Computers Math. Applic.* **27**(3) (1994), 21–25.

[10] F. Harary, *Graph Theory*, Addison-Wesley, 1969.

[11] Y. Saad and M. Schultz, "Data Communication in Parallel Architectures", *Parallel Computing* **11** (1989), 131–150.