GROUP ACTION GRAPHS AND PARALLEL ARCHITECTURES

Fred Annexstein
Marc Baumslag, Arnold L. Rosenberg
Computer and Information Science Department
University of Massachusetts

COINS Technical Report 87-133

GROUP ACTION GRAPHS AND PARALLEL ARCHITECTURES

Fred Annexstein
Marc Baumslag
Arnold L. Rosenberg
Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

ABSTRACT. We develop a mathematical framework that exposes the structural kinship among the *DeBruijn, Shuffle-Exchange, Butterfly*, and *Cube-Connected Cycles* networks, and we illustrate algorithmic benefits that can be gleaned from the exposed relationships. Our framework builds on two algebraically specified genres of graphs: A *group-action graph* (*GAG*, for short) is given by a set V of vertices and a set Π of permutations of V: For each $v \in V$ and each $\pi \in \Pi$, there is an arc labelled π from vertex v to vertex $v\pi$. A *Cayley graph* is a GAG (V, Π), where V is the group $Gr(\Pi)$ generated by Π and where each $\pi \in \Pi$ acts on each $g \in Gr(\Pi)$ by right multiplication. We call the graphs $(Gr(\Pi), \Pi)$ and (V, Π) *associated* graphs. We show that every GAG is a quotient-graph of its associated Cayley graph. By applying such general results, we determine the following.

- The Butterfly network (a Cayley graph) and the DeBruijn graph (a GAG) are associated graphs;
- the Cube-Connected Cycles network (a Cayley graph) and the Shuffle-Exchange graph (a GAG) are associated graphs;
- the n^{th} version of both the Butterfly and the Cube-Connected Cycles share the same underlying group, but have slightly different generator sets Π.

By analyzing the algebraic setting, we delimit, for any Cayley graph G and associated GAG H, a family of algorithms that run as efficiently on H as they do on (the much larger) G.

1. INTRODUCTION

1.1. Background

We develop an algebraic setting for studying certain structural and algorithmic properties of the interconnection networks that underlie parallel architectures. Our study and approach find their origins in three sources.

- Akers and Krishnamurthy [AK1, AK2, AK3] argue that there are significant advantages, relating to both algorithmic efficiency and fault tolerance, that accrue when one designs a parallel architecture so that its underlying interconnection network is highly symmetric. They argue particularly for the level of symmetry coupled with tractability that one finds in *Cayley graphs*, i.e., graphs whose adjacency structure is governed by a *group*.[1] These papers make a strong case for their position, which is strengthened by the well-known advantages of well-known interconnection networks such as the Hypercube, the Butterfly, the Cube-Connected Cycles, and the ring, all of which are Cayley graphs. However, two factors somewhat mitigate the effectiveness of their arguments: First, there are a number of "semi-uniform" interconnection networks that are promoted as vigorously by their proponents as are the just-enumerated Cayley graphs; included among these are the Shuffle-Exchange, the DeBruijn, the X-tree, and the tree networks. Second, well-known group-theoretic results suggest that the "natural" way of defining Cayley-graph-based interconnection networks - which is the way advocated in [AK2] and employed here - is almost certain to yield networks that are truly enormous[2]. Our first motivation here was to find a genre of graph that enjoys many of the algorithmic advantages of Cayley graphs, but at a more moderate cost in terms of the size of the graph. The notion of *group-action graph* that we develop here captures two of the four mentioned "semi-uniform" graphs.

- Erdös [EK1] defines a genre of graph-augmentation problem that is related to the issue just discussed: Given a graph G with maximum vertex-degree[3] d, what is the smallest number of new vertices and edges one must add to G to render it regular of degree d? Solutions to three variants of this problem appear in [AEH, EK1, EK2]. The issues raised in [AK1, AK2, AK3] suggest a similar augmentation problem for producing Cayley graphs. One part of the research reported on here began by studying one approach to this problem, namely by augmenting a given interconnection network in stages, from an undirected graph to a directed graph to a "group-action graph" to a Cayley graph. (See footnote 1.)

- The major shortcoming of the (Boolean) Hypercube as an interconnection network is its high vertex-degrees.[4] This fact has led to the introduction of several bounded-degree "approxi-

[1] Detailed definitions appear in Section 1.2.

[2] Technically, the groups underlying these graphs have large *symmetric* or *alternating* groups as subgroups: The symmetric group on the set S is the collection of all permutations of S, which are $|S|!$ in number; the alternating group on S comprises half of these permutations.

[3] The *degree* of a vertex is its number of neighbors; a graph is *regular* if all vertices have the same degree.

[4] Each vertex of the N-vertex Hypercube has degree $\log N$.

mations" of the Hypercube, most notably the Butterfly and Cube-Connected Cycles (CCC) networks. Ad hoc transformations of these large[5] networks have led to smaller Hypercube-derivatives, most notably the Shuffle-Exchange and DeBruijn networks, that share the size of the Hpercube and that afford one computational efficiency (roughly) equal to that of the Butterfly and CCC, on certain computational tasks. Indeed, many in the parallel architecture community will attest to the computational equivalence of the four networks just mentioned, supporting their assertion by exhibiting a class of parallel algorithms that run efficiently on all Hypercube derivatives (including the four under discussion) that share certain structural characteristics. Roughly speaking, the processors of these networks have addresses containing length-n bit-strings. The algorithms in the class have the property that, for each time step i, all communication is between processors whose addresses differ only in the i^{th} place in their bit-string (but possibly differ in other components of the address as well). Many well-known, efficient algorithms reside in this class, for such tasks as sorting, computing convolutions (using the FFT algorithm [AHU, Ch. 7]), and matrix operations; see [PV, Ul] for details. Our goal was to find a general mathematical setting that would expose the structural and algorithmic relationships among these four interconnection networks, with the hope that such a general framework would help us with the first of our three goals, as well as explain rigorously the asserted "equivalence" of these networks.

Aside: To whet the reader's appetite for the conclusions of Section 3: Figs. 1,2 depict the 3-dimensional DeBruijn and Butterfly networks, respectively. It is not clear at first blush how to map the Butterfly network onto the DeBruijn network in a structure-preserving manner. Fig. 3 depicts schematically how our framework accomplishes this.

The results we report on here can be viewed as progress toward all three goals. We find this progress particularly satisfying since each step in our development is dictated by the structure of the problems studied.

We envision the development here as a first step towards exploring the application of group-theoretic techniques coupled with graph-theoretic techniques in the design and analysis of parallel architectures.

1.2. The Formal Setting

A. Graph-Theoretic Notions

We deal with three levels of graph structure.

- An *undirected graph* G is given by a set V of *vertices* and a set of doubleton subsets of V called *edges*. The vertices in an edge are said to be *adjacent* in G. A *path* in G is a sequence of distinct vertices, $v_1, v_2, \cdots, v_\ell$, with each pair v_i, v_{i+1} adjacent in G. G is *connected* if each pair of vertices in V appear on some path in G.

[5] Each uses $N \cdot \log N$ vertices to simulate the N-vertex Hypercube.

- A *directed graph* (*digraph*, for short) G is given by a set V of *vertices* and a *multi*subset of $V \times V$ called *arcs*.[6] The underlying undirected graph of G is obtained by replacing each arc of G by the corresponding unordered set and disposing of the set when it is not a doubleton. G is *connected* when its underlying undirected graph is. G is *strongly connected* if, for every ordered pair $\langle v, w \rangle$ of vertices of G, there is a *directed path from v to w*, i.e., a sequence of vertices, $v = v_1, v_2, \cdots, v_\ell = w$, with each pair $\langle v_i, v_{i+1} \rangle$ an arc of G.

- A *transformation graph* (*tragraph*, for short) is given by a set V of *vertices* and a set Φ of transformations of V. For each $v \in V$ and each $\phi \in \Phi$, there is an arc labelled ϕ from vertex v to vertex $v\phi$.[7] The digraph underlying the tragraph G is obtained by erasing the labels from the arcs of G and removing any resulting multiple arcs. A tragraph is connected (resp., strongly connected) just when its underlying digraph is.

Tragraphs are well-known in semigroup theory as *operands* [CP]; the third author studied strongly connected tragraphs extensively, under the name *data graphs* ([Ro1, Ro2] being the most relevant for the present study).

We now define the specific genres of graphs that occupy our attention.

A *group-action graph* (*GAG*, for short) is a tragraph (V, Φ) for which each transformation in the set Φ is a *permutation* of the set V: For mnemonic emphasis, we henceforth denote the transformation-set Π.

Clearly, every GAG has underlying it a digraph all of whose indegrees and outdegrees are equal[8]. The proof in [Ko] that every such "two-way regular" digraph can be 2-factorized (cf. [BM]) shows that this property is sufficient also, in that it can be used to prove the following.

Proposition 1 *Given a digraph $G = (V, E)$, there is a labelling of each arc of G with a permutation of the vertex-set V in a way that makes G a GAG if, and only if, there is a constant c such that every vertex of G has both indegree c and outdegree c.*

A little background on groups is necessary before we begin to discuss the highly uniform graphs that our study focusses on.

B. Group-Theoretic Notions

A *group* is given by a set S, together with an associative binary *multiplication* on S (denoted by a centered dot) that has an *identity* - an $e \in S$ for which $s \cdot e = e \cdot s = s$, for all $s \in S$ and *inverses* - for each $s \in S$, an element $t \in S$ for which $s \cdot t = t \cdot s = e$.

A *Cayley graph* (or, *group graph*) is a GAG (V, Π), where V is the group $Gr(\Pi)$ generated by Π, and where each $\pi \in \Pi$ acts on $Gr(\Pi)$ by right multiplication, so that $\pi \in \Pi$ "leads" vertex

[6] By using *multi*subsets of $V \times V$, we allow multiple arcs, i.e., several arcs connecting a given pair of vertices.

[7] $v\phi$ denotes the image of v under the transformation ϕ.

[8] The *indegree* (resp., *outdegree*) of a vertex v of G is the number of arcs of G having v as their second (resp., their first) element.

$g \in Gr(\Pi)$ to vertex $g \cdot \pi$. We call Π the set of *generators* of the group $Gr(\Pi)$. We denote by $Cay(\Pi)$ the Cayley graph induced by the set Π of permutations.

We call the Cayley graph $(G(\Pi), \Pi)$ and the GAG (V, Π) *associated* graphs.

There is clearly a "natural" way to construct a Cayley graph from any GAG (V, Π), namely, $Cay(\Pi)$. Less obviously, there is a "natural" way to construct GAGs from any Cayley graph (usually more than one).

Given any subgroup $\mathcal{H}$ of a group $\mathcal{G}$ (i.e., a subset of $\mathcal{G}$, that is a group under the multiplication in $\mathcal{G}$), the *quotient* of $\mathcal{G}$ by $\mathcal{H}$, denoted $\mathcal{G}/\mathcal{H}$, is the collection of all *right cosets of* $\mathcal{H}$ *in* $\mathcal{G}$: For each $g \in \mathcal{G}$, the *right coset of* $\mathcal{H}$ *containing* g, denoted $\mathcal{H}g$, is the set of all left multiples $h \cdot g$ of g by elements $h \in \mathcal{H}$. It is a standard result that the cosets of $\mathcal{H}$ *partition* $\mathcal{G}$ into blocks of equal size.

These notions yield the "natural" construction of GAGs from Cayley graphs.

Let $\mathcal{H}$ be a subgroup of the group $\mathcal{G} = Gr(\Pi)$. The *coset graph of* $\mathcal{G}$ *with respect to* $\mathcal{H}$ *and* Π, denoted $Cos(\mathcal{G}; \mathcal{H}; \Pi)$, is the GAG $(\mathcal{G}/\mathcal{H}, \Pi)$ whose vertex-set is the set of right cosets of $\mathcal{H}$ in $\mathcal{G}$, and whose arcs are given by the action of the elements of Π viewed as permutations of $\mathcal{G}/\mathcal{H}$, this action being defined by right multiplication: for $g \in G$ and $\pi \in \Pi$,

$$(\mathcal{H}g)\pi = \mathcal{H}(g \cdot \pi)$$

Our study builds on the fact that a group can be viewed both as an abstract algebraic structure and as a collection of permutations – multiplication in the latter view being functional composition. This important fact is formalized in Cayley's Theorem [Ha]:

Proposition 2 [Ha] *Every group is isomorphic to a set of permutations.*

Let $\mathcal{G}$ be a group of permutations of the set S.

- $\mathcal{G}$ *acts transitively on* S (*is transitive*, for short) if, for all $s, t \in S$, there is a $g \in \mathcal{G}$ such that $sg = t$.
- For each $s \in S$, the *stabilizer* of s in $\mathcal{G}$, denoted $St(\mathcal{G}; s)$, is the set of all permutations in $\mathcal{G}$ that *fix* s, i.e., for which $s\pi = s$. It is a standard result that $St(\mathcal{G}; s)$ is a subgroup of $\mathcal{G}$.

A *cyclic group* is a group whose underlying set is (isomorphic to) $Z_d =_{\text{def}} \{0, 1, \cdots, d-1\}$ and whose multiplication is (isomorphic to) addition modulo d. We denote the d-element cyclic group by Z_d, allowing context to distinguish between the group and its underlying set.

The *wreath product* of cyclic group Z_d by cyclic group Z_n, denoted $Z_d \otimes_{wr} Z_n$, is a group of permutations of the set

$$Z_d^n =_{\text{def}} \{0, 1, \cdots, d-1\}^n.$$

Each element of $Z_d \otimes_{wr} Z_n$ is an $(n+1)$-tuple

$$\pi = \langle \alpha; \beta_0, \beta_1, \cdots, \beta_{n-1} \rangle$$

where $\alpha \in Z_n$ and each $\beta_i \in Z_d$. The action of the permutation π on the element $(\delta_0, \delta_1, \cdots, \delta_{n-1}) \in Z_d^n$ consists of a modulo-d vector addition of $(\beta_0, \beta_1, \cdots, \beta_{n-1})$, followed by a sequence of α left-cyclic shifts:

$$(\delta_0, \delta_1, \cdots, \delta_{n-1})\langle \alpha; \beta_0, \beta_1, \cdots, \beta_{n-1} \rangle = (\delta_\alpha + \beta_\alpha, \cdots, \delta_{n-1} + \beta_{n-1}, \delta_0 + \beta_0, \cdots, \delta_{\alpha-1} + \beta_{\alpha-1})$$

Multiplication in a wreath product is composition of permutations.

1.3. Synopsis of Our Results

Not only can one determine the structure of a Cayley graph $Cay(\Pi)$ from the structure of its associated GAG (V, Π) (by looking at the multiplication table of $Gr(\Pi)$), one can also determine the structure of the GAG from the structure of its associated Cayley graph (if the group is presented as a group of permutations of V).

Theorem 1. *Every connected GAG* (V, Π) *is isomorphic to the coset graph* $Cos(Gr(\Pi); \mathcal{H}; \Pi)$, *where* $\mathcal{H}$ *is the stabilizer of any* $v \in V$.

Thus, every GAG is a factor graph of its associated Cayley graph. We use this result to derive a formal, rigorous version of the old saw, "The Butterfly, Cube-Connected Cycles, Shuffle-Exchange, and DeBruijn networks are equivalent as interconnection networks." Specifically, we show that:

- the Butterfly network (a Cayley graph) and the DeBruijn graph (a GAG) are associated graphs;
- the Cube-Connected Cycles network (a Cayley graph) and the Shuffle-Exchange graph (a GAG) are associated graphs;
- the associated group of the n^{th} version of both the DeBruijn and Shuffle-Exchange graphs is the wreath product $Z_2 \otimes_{wr} Z_n$, but their associated Cayley graphs (the Butterfly and CCC, respectively) have slightly different generator sets Π.

Adding to their structural interest, these correspondences have algorithmic consequences:

Theorem 2. (Informal Version) *One can automatically translate any "levelled" algorithm for a given Cayley graph into an equally efficient algorithm for its associated GAG.*

This result is proved via an algorithm that generalizes and strengthens a simulation algorithm described in [Ul]. Noting the structural similarity between the Shuffle-Exchange and Butterfly networks, Ullman shows that any so-called "normal" algorithm[9] that runs on the Butterfly can be modified to run on the Shuffle-Exchange in twice the time. Using the relationships we develop in Theorem 1 between Cayley graphs and their associated GAGs, we derive Theorem 2, one of

[9] We use the term "levelled" here, rather than "normal", to avoid the possibility of confusion with the notion of normality in group theory.

whose consequences is that "levelled" Butterfly algorithms run on the DeBruijn network *with no slowdown*.

Section 2 is devoted to proving Theorems 1 and 2, in the abstract formulation of this subsection. Section 3 develops the algebraic structure of the four interconnection networks that are our special focus, and specializes our abstract results to these networks.

2. MAIN RESULTS: ABSTRACT VERSION

2.1. A Structural Relationship between GAGs and Coset Graphs

The group structure underlying GAGs automatically induces certain uniformities in the underlying graphs.

Lemma 1 *Every connected GAG is strongly connected. It follows that, if* (V, Π) *is a connected GAG, then* $Gr(\Pi)$ *is a transitive group.*

Proof. By Proposition 1, the digraph underlying a connected GAG is a connected digraph with equal indegree and outdegree at each vertex. By Exercise 10.3.2 of [BM], every such digraph contains an *Eulerian tour* (i.e., a closed directed path traversing each arc exactly once). This tour yields a directed path between any pair of vertices, whence the claim of strong connectivity. To see that $Gr(\Pi)$ is transitive, observe that for any pair of vertices (v, w), the sequence of labels on a directed path from v to w defines (via composition of permutations) a permutation in $Gr(\Pi)$ which maps v to w. □

We are now in a position to prove that every connected GAG $G = (V, \Pi)$ is a coset graph, within the strong context of Theorem 1. Since Lemma 1 assures us that $\mathcal{G} = Gr(\Pi)$ is a transitive permutation group, we can simplify our setting slightly. We are concerned with stabilizers of elements of V in $\mathcal{G}$. In a transitive permutation group, all stabilizers are conjugate[10], hence all isomorphic. Thus, we can refer to *the stabilizer subgroup of* $\mathcal{G}$, written $St(\mathcal{G})$, without focussing on which $v \in V$ we are fixing. Our formal version of Theorem 1 thus becomes:

Theorem 1 *Every connected GAG* (V, Π) *is isomorphic (*as a tragraph*) to the coset graph*

$$Cos(Gr(\Pi);\ St(Gr(\Pi));\ \Pi).$$

Proof. Let the GAG $G = (V, \Pi)$ and the group $\mathcal{G} = Gr(\Pi)$ be as in the statement of the Theorem. Pick an arbitrary $v \in V$,[11] and let $\mathcal{H} = St(\mathcal{G}; v)$. Establish the following mapping μ from cosets in $\mathcal{G}/\mathcal{H}$ to V: for $g \in \mathcal{G}$,

[10] Subgroups $\mathcal{H}_1$ and $\mathcal{H}_2$ of $\mathcal{G}$ are *conjugate* if every right coset of $\mathcal{H}_1$ by any $g \in \mathcal{G}$ is a left coset of $\mathcal{H}_2$ by the same element; symbolically, $\mathcal{H}_1 g = g\mathcal{H}_2$ for all $g \in \mathcal{G}$..

[11] The specific v chosen is immaterial since $\mathcal{G}$ is transitive.

$$(\mathcal{H}g)\mu = w \text{ if, and only if, } (vh)g = v(h \cdot g) = w \text{ for each } h \in \mathcal{H}.$$

μ thus associates vertex w of G with that coset of $\mathcal{G}/\mathcal{H}$ comprising the permutations in $\mathcal{G}$ that map v to w. Since $\mathcal{G}$ is a group of permutations, and since $\mathcal{H}$ is the stabilizer of v in $\mathcal{G}$, the mapping μ is well-defined and one-to-one; moreover, since $\mathcal{G}$ is transitive (by Lemma 1), the mapping μ is also onto. Therefore, once we show that μ preserves (labelled) arcs, we shall be done.

Say first that, in the GAG, there is an arc labelled π from vertex u to vertex w. By definition, then, π is a permutation in Π for which $u\pi = w$. Now, for every permutation $g \in \mathcal{G}$ that maps v to u (i.e., $vg = u$), the permutation $g \cdot \pi$ maps v to $u\pi = w$, via the equations

$$w = u\pi = (vg)\pi = v(g \cdot \pi).$$

It follows that there is an arc labelled π from vertex $u\mu$ to vertex $w\mu$ in the coset graph.

Finally, say that, in the coset graph, there is an arc labelled π from vertex $\mathcal{H}f$ to vertex $\mathcal{H}g$ $(f, g \in \mathcal{G})$. By definition, then, we have the equation

$$\mathcal{H}g = \mathcal{H}(f \cdot \pi)$$

on the right cosets of $\mathcal{H}$. It follows that for each permutation $h \in \mathcal{H}$, we have

$$v(h \cdot g) = (vh)g = vg = v(f \cdot \pi) = (vf)\pi$$

since $\mathcal{H}$ is the stabilizer of v in $\mathcal{G}$. But the latter equations imply that there is an arc labelled π from vertex vf to vertex vg in the GAG. □

We have already noted that the choice of the stabilizer $\mathcal{H}$ in Theorem 1 does not affect the structure of the GAG, but it does change the correspondence between the right cosets of $\mathcal{H}$ in $\mathcal{G}$ and vertices of G.

The next two corollaries of Theorem 1 exhibit useful relationships between a GAG $G = (V, \Pi)$ and its induced Cayley graph $Cay(\Pi)$. The first result indicates that certain simple structures in G replicate in $Cay(\Pi)$ with the multiplicity suggested by Theorem 1's structural characterization of GAGs as coset graphs.

Corollary 1 *Each directed tree*[12] T *that is a subgraph of the GAG* (V, Π) *appears with multiplicity* $|\,St\,(Gr(\Pi))|$ *as a subgraph of* $Cay(\Pi)$.

Proof. Let the directed tree T be a subgraph of the GAG $G = (V, \Pi)$. The mapping μ in the proof of Theorem 1 associates each vertex v of T with a unique right coset $\mathcal{H}g$, where $g \in Gr(\Pi)$ and where $\mathcal{H} = St(Gr(\Pi); w)$ for some fixed but arbitrary $w \in V$. As we noted earlier, all of the candidate cosets have common size $|\,St\,(Gr(\Pi))|$.

[12] A *tree* is a connected undirected graph that has a unique path between every pair of vertices; a *leaf* in the tree is a vertex of unit degree. A *directed tree* is a digraph whose underlying graph is a tree with a designated *root* vertex, all of whose arcs are oriented from root to leaf.

Consider now an arbitrary arc labelled $\pi \in \Pi$ from vertex u to vertex w in G. In $Cay(\Pi)$, the permutation π induces a bijection (via right multiplication) between the cosets $\mu^{-1}(u)$ and $\mu^{-1}(w)$. Thus, each arc in T spawns $|St(Gr(\Pi))|$ distinct arcs in $Cay(\Pi)$. Since no two arcs in the tree T enter the same vertex, this replication of arcs suffices to establish the result. □

Note that one cannot extend Corollary 1 very far, since a cycle in the GAG G may not result in a cycle in the induced Cayley graph: the "initial" and "final" arcs in the Cayley graph will start and end (respectively) in the same coset, but not necessarily at the same vertex in the coset.

Theorem 1 also affords us a general upper bound on the diameter[13] of a Cayley graph, in terms of the diameter of an associated GAG.

Corollary 2 *The diameters of the connected GAG $G = (V, \Pi)$ and its associated Cayley graph $\Gamma = Cay(\Pi)$ satisfy the relation*

$$\mathrm{diam}(\Gamma) \leq \mathrm{diam}_{\mathcal{H}}(\Gamma) + \mathrm{diam}(G),$$

where $\mathcal{H} = St(Gr(\Pi))$ and $\mathrm{diam}_{\mathcal{H}}(\Gamma)$ is the diameter of the subgraph of Γ induced on the vertices of $\mathcal{H}$.[14] *Moreover, if $\mathcal{H}$ is generated by a proper subset Ψ of Π, then*

$$\mathrm{diam}(\Gamma) \leq \mathrm{diam}(Cay(\Psi)) + \mathrm{diam}(G).$$

Proof. At an intuitive level, we are asserting that one can travel from vertex u to vertex v in Γ by using copies of GAG-arcs to travel from the coset containing u to the coset containing v, then using copies of the arcs that stay within $\mathcal{H}$, hence within each coset defined by $\mathcal{H}$, to get to v.

More formally, we note that, because of vertex-transitivity, the diameter of Γ is just the maximum distance of any vertex from the identity vertex e (the identity of the underlying group $Gr(\Pi)$, that is). A path between e and any other $g \in Gr(\Pi)$ can be found as follows. Say that g is in the coset $\mathcal{H}a$ in $Gr(\Pi)/\mathcal{H}$. As in the proof of Theorem 1, for each $g \in Gr(\Pi)$ and each $\pi \in \Pi$, Γ contains $|\mathcal{H}|$ parallel arcs labelled π connecting the vertices in coset $\mathcal{H}g$ of $Gr(\Pi)/\mathcal{H}$ (in a one-to-one fashion) to the vertices in coset $\mathcal{H}(g \cdot \pi)$: These parallel arcs correspond to the arc labelled π in the GAG G, from vertex $\mathcal{H}g$ to vertex $\mathcal{H}(g \cdot \pi)$. One follows a path of such inter-coset arcs in Γ, of length at most $\mathrm{diam}(G)$, starting at vertex e (which is in coset $\mathcal{H}$) and ending at some vertex in coset $\mathcal{H}a$. In general, inter-coset arcs will not take us directly to g, so we must then follow a path of *intra-coset* arcs to attain vertex g, all the while staying within coset $\mathcal{H}a$. The traversed path is guaranteed to work and has length at most $\mathrm{diam}_{\mathcal{H}}(\Gamma) + \mathrm{diam}(G)$, whence the result. The second inequality follows by the same reasoning, when $\mathcal{H}$ is generated by a subset of Π. □

[13] The *distance* between vertices v and w of a digraph G is the length of the shortest directed path from v to w; the *diameter* of G is the longest of these shortest paths, for any pair of vertices.

[14] The *induced subgraph* comprises all vertices of $\mathcal{H}$, plus all arcs of Γ that connect vertices of $\mathcal{H}$.

2.2. Algorithmic Consequences of the Structural Relationship

The development in Section 2.1 has algorithmic consequences that often allow one to trade significant savings in the number of processors in one's array for a modest increase in computing time. For a special class of algorithms, which includes certain algorithms for sorting and computing the FFT, one can obtain the savings in processor count *with no time loss*. We now describe the algorithmic setting, which generalizes an analogous discussion in [Ul].

Say that we are given an algorithm A that runs in synchronous mode on a parallel architecture whose interprocessor communication structure is given by a digraph G. Say further that Algorithm A runs on G in the following format:

> There is a partition of the set of vertices of G (which are the processors of the array) into sets $V_1, V_2, \ldots, V_\ell$, such that, at each moment of time, the active set of processors involves at most one processor from each set V_i.

We call each set V_i a *block of the graph* G, and we call Algorithm A an (ℓ-)*block-structured algorithm*. Consider now the following modification of the scenario just described.

- Label the vertices/processors of G in any way that assigns a different label to each processor in each block V_i; clearly $N =_{\text{def}} \max_i |V_i|$ labels suffice.
- Construct the ℓ-vertex graph/processor array G' that has a vertex v_i for each block V_i in the partition of G, and that has an arc from vertex v_a to vertex v_b just when, in G, some vertex in block V_a has an arc to some vertex in block V_b. Give each vertex v_i of G' the capability to simulate each processor in the block V_i of G. (This is easy with arrays of identical processors.)
- Modify Algorithm A to obtain Algorithm A' that operates as follows. Each message generated by Algorithm A' is a message generated by Algorithm A, augmented with the label (address) of the processor of G that the message is intended for.
- Run Algorithm A' on graph G' as follows.
  - If an initial message M of Algorithm A would go to processor v in block V_i of G, then Algorithm A' tags Message M with the label of v and sends it to processor v_i of G'.
  - When a processor of G' completes a task of Algorithm A', it tags each message M that it has generated with the label of the processor v (in block V_i) of G that Algorithm A would send the message to on G. It then sends the message to processor v_i of G'.
- A processor v_i of G' uses the label attached to incoming messages to determine which processor in block V_i of G to simulate during a given step of Algorithm A'. The block-oriented character of Algorithm A guarantees that a processor of G' is never asked to simulate more than one processor of G at a time.

It is transparent that Algorithm A' is functionally equivalent to Algorithm A; moreover, the only overhead for running the former algorithm to simulate the latter resides in the process of appending, sending, and decoding the processor-labels, which can be assumed to be bit-strings of length at most $\log N$. This overhead allows us to simulate a large processor array with an ℓ-processor array.

When the graph G is a Cayley graph $Cay(\Pi)$, and the graph G' is a coset graph $Cos(Gr(\Pi); \mathcal{H}; \Pi)$ of G by the subgroup $\mathcal{H}$, then the scenario just described is often simplified somewhat, for each block V_i of G has the same number of processors, $|\mathcal{H}|$. Thus, the effect of Algorithm A on array G is obtained on a processor array of size $|G|/|\mathcal{H}|$. In this case, we say that Algorithm A is *$\mathcal{H}$-blocked.*

In certain instances, we can do even better. Consider, for illustration, executing the FFT algorithm on the Butterfly network (a Cayley graph whose structure matches the data dependencies of the algorithm; cf. Section 3.1.B and [AHU, Ch. 7]). This algorithm runs on the network level by level (using the natural levelling of the network): At each time t, each processor at level t of the network takes in inputs x_1 and x_2 on its two input ports, using the source input ports to distinguish x_1 from x_2; it computes two linear functions $L_{1,t}(x_1, x_2)$ and $L_{2,t}(x_1, x_2)$ of the inputs; it sends $L_{1,t}$ out on its first output port and $L_{2,t}$ out on its second output port. Thus, at any specific moment, only one level of processors in the network is active. One sees that this computation is so carefully choreographed that no addressing mechanism is needed to determine what a processor is to do: A processor can determine what role to play – i.e., for which t to compute $L_{1,t}$ and $L_{2,t}$ merely by *keeping track of the time.* Generalizing from this example, we call an $\mathcal{H}$-blocked algorithm for the Cayley graph G *orchestrated* if one can label each processor in each block of G (i.e., coset of $\mathcal{H}$) with a set of time-stamps that indicate the times when that processor is active while executing the algorithm, *independent of the input data.* As with our example, one can execute an orchestrated blocked algorithm on the coset graph using only a clock (either global or one per processor) to tell each processor of the coset graph when it is to play which role.

In the very special case of our example of the FFT algorithm on the Butterfly network, we encounter the potential for even further simplification: In the FFT algorithm, the differences among the various linear functions $L_{i,t}$, as t varies, reside in a parameter ω_t that enters into the computation of $L_{i,t}$. Since each $\omega_t = \omega_{t-1}^2$, a further algorithmic simplification is possible: If we have each processor square its current parameter before computing its linear functions, then we shall have just two fixed linear functions $L_i(x_1, x_2; \omega_{current})$, $i = 1, 2$, that are computed by every processor in the Butterfly. In this case, there is no need for a processor of a coset graph to maintain any information about its "identity": *Every processor, in every block/coset, performs the same computation at every step as does every other processor.* In the case of such *oblivious* algorithms, therefore, we do not need any global or local clocks, and we do not need to devote any time to processing addresses or time-stamps: We save a large factor in hardware *at no extra cost in computation time*[15].

We close this section with a formal analog of Theorem 2 whose proof is contained in the preceding discussion.

Theorem 2 *Let $Cay(\Pi)$ be a Cayley graph, let $\mathcal{H}$ be a subgroup of $Gr(\Pi)$, and let A be an $\mathcal{H}$-blocked*

[15] We do, however, lose the ability to pipeline computations (one per level on the Butterfly).

algorithm for $Cay(\Pi)$.
(a) *Algorithm A can be run on the coset graph* $G = Cos(Gr(\Pi); \mathcal{H}; \Pi)$, *slowed down by the factor* $O(\log|\mathcal{H}|)$.
(b) *If Algorithm A is orchestrated, then t steps of the Algorithm can be run on the coset graph G in* $O(t \log t)$ *steps.*
(c) *In either of the previous circumstances, if the processors of* $Cay(\Pi)$ *operate on "large" quantities, then the slowdown on the coset graph G is only* $O(1)$.
(d) *If Algorithm A is oblivious, then it can be run on the coset graph G with no time loss.*

As a final remark in this section, we note that the Cayley graph $Cay(\Pi)$ can be enormous compared to its associated GAG (V, Π). Even if Π consists of two permutations, one of which cyclically permutes V and one of which switches two elements of V while holding all others fixed - so that the undirected graph underlying the GAG (V, Π) has only $|V|$ edges – the group $Gr(\Pi)$ can contain $|V|!$ elements. Although we do not know of any Cayley graph - GAG - algorithm matchups that apply Theorem 2 to such an extreme situation, it is conceivable that such do exist.

3. MAIN RESULTS: CONCRETE VERSION

In this section we demonstrate the usefulness of the development in the preceding section, by applying it to four families of graphs that are benchmarks among interconnection networks for parallel architectures:

- the *Butterfly* network
- the *Cube-Connected Cycles* network
- the *DeBruijn* network
- the *Shuffle-Exchange* network

All four families are bounded-degree "approximations" to the Hypercube network. We prove that the first two are families of Cayley graphs, both having the same wreath products of cyclic groups as underlying groups, but with slightly different generator sets; we prove that the second two are families of coset graphs of the former two families. Thus we characterize precisely and rigorously the structural similarities and differences among these families. We then discuss the implications of Theorem 2 for the correspondences we have exposed.

At virtually no extra cost, we establish our results for generalized versions of the four studied networks.

3.1. DeBruijn and Butterfly Networks

A. DeBruijn Graphs

Let d, n be positive integers. The *base-d n-dimensional DeBruijn graph* $\Delta(n;d)$ is the digraph whose vertices comprise the set Z_d^n, and whose arcs connect each vertex $\alpha x \in Z_d^n$, where $\alpha \in Z_d$ and $x \in Z_d^{n-1}$, to each vertex of the form $x\beta \in Z_d^n$ for some $\beta \in Z_d$; see Fig. 1, where the conventional, base-2, DeBruijn graph is depicted, and [dB, Fr, ISO].

By Proposition 1, every DeBruijn graph can be arc-labelled so as to be a GAG. One way to do this yields the correspondences we seek. For each $\beta \in Z_d$, define the permutation $\pi[\beta;d]$ of Z_d^n (n being clear from context) by

$$(\alpha x)\pi[\beta;d] = x(\alpha + \beta(\text{mod } d)),$$

for each $\alpha \in Z_d$ and $x \in Z_d^{n-1}$.[16] Label each arc of $\Delta(n;d)$ of the form $(\alpha x, x\beta)$ with the permutation $\pi[\beta - \alpha(\text{mod } d);d]$. We leave to the reader the easy verification that the described arc-labelling renders $\Delta(n;d)$ a GAG.

Let $\Pi_d^\Delta =_{\text{def}} \{\pi[\beta;d] \mid \beta \in Z_d\}$.

Lemma 2 *For all d, $Gr(\Pi_d^\Delta)$ is isomorphic to the wreath product $Z_d \otimes_{wr} Z_n$.*

Proof. Note first that for each base d and integer $\beta \in Z_d$, the permutation $\pi[\beta;d] \in \Pi_d^\Delta$ is equivalent to a modulo-d vector addition of $(\beta,0,\cdots,0)$ to the argument string/vector, followed by a one-place-left cyclic shift of the digits of the argument; $\pi[\beta;d]$ is, thus, equivalent to the permutation

$$\langle 1;\beta,0,\cdots,0\rangle$$

in $Z_d \otimes_{wr} Z_n$. It follows, therefore, that $Gr(\Pi_d^\Delta)$ is a subgroup of $Z_d \otimes_{wr} Z_n$.

Next, consider an arbitrary permutation

$$\pi = \langle \alpha;\beta_0,\beta_1,\cdots,\beta_{n-1}\rangle$$

in $Z_d \otimes_{wr} Z_n$. As we noted in Section 1.2.B, the action of π on an element of Z_d^n consists of a modulo-d vector addition of $(\beta_0,\beta_1,\cdots,\beta_{n-1})$ to the argument, followed by α left-cyclic shifts. The action of π is, thus, equivalent to the action of the product[17]

$$\pi[\beta_0;d]\pi[\beta_1;d]\cdots\pi[\beta_{n-1};d](\pi[0;d])^\alpha$$

of permutations from Π_d^Δ: the first n permutations effect the vector addition, while the last α effect the cyclic shift. It follows, therefore, that $Z_d \otimes_{wr} Z_n$ is a subgroup of $Gr(\Pi_d^\Delta)$.

The Lemma follows. □

[16] For all d, the permutation $\pi[0;d]$ is just a cyclic-shift of the argument string. The permutation $\pi[0;2]$ is termed a *perfect shuffle*, and the permutation $\pi[1;2]$ is termed a *shuffle-exchange*.

[17] In the product, $(\pi[0;d])^\alpha$ denotes a sequence of α instances of $\pi[0;d]$.

Lemma 3 *For all d, n, the base-d, n-dimensional DeBruijn graph $\Delta(n;d)$ is isomorphic to the Coset graph*

$$Cos(\mathcal{G}; \mathcal{H}; \Pi_d^{\Delta})$$

where $\mathcal{G} = Z_d \otimes_{wr} Z_n$ and $\mathcal{H} = \{0\} \otimes_{wr} Z_n$.

Proof. Since $\Delta(n;d)$ is a connected GAG, the Lemma will follow from Theorem 1, once we determine the stabilizer of Z_d^n in $Z_d \otimes_{wr} Z_n$. By Theorem 1, it will suffice to determine the stabilizer of the element $(0,0,\cdots,0) \in Z_d^n$, which is transparently $Gr(\pi[0;d|)$ (i.e., all possible shifts, with no additions). Using reasoning analogous to that in the proof of Lemma 2, one verifies that $Gr(\pi[0;d|)$ is isomorphic to the subgroup $\{0\} \otimes_{wr} Z_n$ of $Z_d \otimes_{wr} Z_n$. □

B. Butterfly Networks

Let d, n be a positive integer. The *base-d n-level Butterfly graph* $B(n;d)$ has vertex-set

$$V_{n;d} = Z_n \times Z_d^n.$$

The subset $V_{n;d}^{(\ell)} = \{\ell\} \times Z_d^n$ of $V_{n;d}$ $(0 \le \ell < n)$ is the ℓ^{th} *level* of $B(n;d)$. The edges of $B(n;d)$ form *d-butterflies* (or, copies of the complete bipartite graph $K_{d,d}$) between consecutive levels of vertices, with wraparound in the sense that level 0 is identified with level n. Each butterfly connects each vertex

$$\langle \ell,\ \beta_0\beta_1\cdots\beta_{\ell-1}\alpha\beta_{\ell+1}\cdots\beta_{n-1}\rangle$$

on level ℓ of $B(n;d)$ $(0 \le \ell < n$; α and each β_i in $Z_d)$ with all vertices

$$\langle \ell+1 (\text{mod } n),\ \beta_0\beta_1\cdots\beta_{\ell-1}\omega\beta_{\ell+1}\cdots\beta_{n-1}\rangle$$

on level $\ell+1 (\text{mod } n)$ of $B(n;d)$, for all $\omega \in Z_d$.[18]

There is a natural way to turn $B(n;d)$ into a tragraph whose arcs are labelled with permutations from Π_d^{Δ}. First, we form the *directed* version $\vec{B}(n;d)$ of $B(n;d)$, by orienting each edge of $B(n;d)$ from level ℓ to level $\ell+1$ (before reducing $\ell+1$ modulo d). Next, we form the *tragraph* version $\tilde{B}(n;d)$ of $\vec{B}(n;d)$, as follows. For each $\ell \in Z_n$ and $\alpha, \omega \in Z_d$, we label the arc from vertex

$$\langle \ell,\ \beta_0\beta_1\cdots\beta_{\ell-1}\alpha\beta_{\ell+1}\cdots\beta_{n-1}\rangle$$

to vertex

$$\langle \ell+1 (\text{mod } n),\ \beta_0\beta_1\cdots\beta_{\ell-1}\omega\beta_{\ell+1}\cdots\beta_{n-1}\rangle$$

with the permutation $\pi[\omega - \alpha (\text{mod } d);\ d]$; see Fig. 2.

Lemma 4 *For all d, n, the base-d n-level Butterfly network $\tilde{B}(n;d)$ is isomorphic (*as a tragraph*) to the Cayley graph $Cay(\Pi_d^{\Delta})$.*

[18] For $d = 2$, $B(n;2)$ can be viewed as the *FFT network* with the input and output vertices (i.e., the top and bottom levels) identified.

Proof. We show that the natural correspondence between vertices of $\tilde{B}(n;d)$ and vertices of $Cay(\Pi_d^{\Delta})$ yields the desired isomorphism. By Lemma 2, the latter set of vertices comprises just the elements of $Z_d \otimes_{wr} Z_n$. Let each vertex

$$\langle \ell,\ \beta_0\beta_1\cdots\beta_{n-1}\rangle$$

($\ell \in Z_n$; each $\beta_i \in Z_d$) of $\tilde{B}(n;d)$ correspond to the element

$$\langle \ell;\ \beta_0,\beta_1,\cdots,\beta_{n-1}\rangle$$

of $Z_d \otimes_{wr} Z_n$ (cf. Section 1.2.B). Since this mapping is well-defined and onto, it is one-to-one also. To complete the proof, we need only verify that our correspondence preserves (labelled) arcs. To simplify exposition in this verification, let us agree that all addition in the remainder of this proof is modulo d.

Every arc labelled $\pi[\gamma;d] \in \Pi_d^{\Delta}$ in $Cay(\Pi_d^{\Delta})$ leads from a vertex

$$\varsigma = \langle \ell;\ \beta_0,\beta_1,\cdots,\beta_{n-1}\rangle$$

to vertex $\varsigma \cdot \pi[\gamma;d]$. To determine the image of this latter vertex in $\tilde{B}(n;d)$, we must consider the action of $\varsigma \cdot \pi[\gamma;d]$ on an arbitrary vector $(\delta_0,\delta_1,\cdots,\delta_{n-1})$, each $\delta_i \in Z_d$. By definition, this action consists of

- addition of the vector $(\beta_0,\beta_1,\cdots,\beta_{n-1})$ followed by a sequence of ℓ left-cyclic shifts,

which is the action of ς, followed by

- addition of the vector $(\gamma,0,0,\cdots,0)$ followed by a single left-cyclic shift,

which is the action of $\pi[\gamma;d]$. This is easily seen to be identical to the action of the permutation

$$\langle \ell+1;\ \beta_0,\beta_1,\cdots,\beta_{\ell-1},\beta_\ell+\gamma,\beta_{\ell+1},\cdots,\beta_{n-1}\rangle$$

of $Z_d \otimes_{wr} Z_n$. There is, thus, an arc labelled $\pi[\gamma;d]$ in $\tilde{B}(n;d)$, from the vertex corresponding to ς to the vertex corresponding to $\varsigma \cdot \pi[\gamma;d]$.

Every arc labelled $\pi[\gamma;d]$ in $\tilde{B}(n;d)$ leads from a vertex

$$v = \langle \ell,\ \beta_0\beta_1\cdots\beta_{\ell-1}\beta_\ell\beta_{\ell+1}\cdots\beta_{n-1}\rangle$$

to vertex

$$\langle \ell+1,\ \beta_0\beta_1\cdots\beta_{\ell-1}(\beta_\ell+\gamma)\beta_{\ell+1}\cdots\beta_{n-1}\rangle$$

By our correspondence, this latter vertex corresponds to the element

$$\eta = \langle \ell+1;\ \beta_0,\beta_1,\cdots,\beta_{\ell-1},(\beta_\ell+\gamma),\beta_{\ell+1},\cdots,\beta_{n-1}\rangle$$

of $Z_d \otimes_{wr} Z_n$. If we let ς denote the element of $Z_d \otimes_{wr} Z_n$ corresponding to vertex v, then reasoning similar to that in the previous paragraph verifies that $\eta = \varsigma \cdot \pi[\gamma; d]$. This verifies that labelled arcs between vertices in $\tilde{B}(n; d)$ betoken like-labelled arcs between the corresponding vertices of $Cay(\Pi)$. □

Lemmas 2, 3, and 4 establish our first sought concrete correspondence; see Fig. 3.

Theorem 3 *For all d, n, the base-d n-dimensional DeBruijn graph $\Delta(n; d)$ is a coset graph of the base-d n-level Butterfly network $\tilde{B}(n; d)$.*

Our development to this point allows us to infer, using Corollary 1, that Butterfly networks contains a lot of disjoint large trees, thus yielding a simple algebraic proof of a combinatorial result from [BCHLR].

Corollary 3 *The base-d n-level Butterfly network $\tilde{B}(n; d)$ contains n disjoint copies of the height-$(n-1)$ complete d-ary tree.*

Proof. It is trivial to verify that the DeBruijn graph $\Delta(n; d)$ contains the height-$(n-1)$ complete d-ary tree as a subgraph. The result, therefore, follows directly from Theorem 3, Lemma 3, and Corollary 1. □

Finally, the advertised algorithmic consequences of Theorem 3 follow from Theorem 2 coupled with an analysis of the structure of the cosets of the group[19] $\{0\} \otimes_{wr} Z_n$ in the group $Z_d \otimes_{wr} Z_n$. It is not hard to verify that each such coset has the form $\{\ell\} \otimes_{wr} Z_n$, for some $\ell \in Z_n$. It follows that the ℓ^{th} such coset corresponds, using the correspondence of Lemma 4, to level ℓ of $\tilde{B}(n; d)$. The reader can readily supply the details.

3.2. Shuffle-Exchange and CCC Networks

A. Shuffle-Exchange Graphs

Let d, n be positive integers. The *base-d n-dimensional Shuffle-Exchange graph* $\Sigma(n; d)$ is the GAG whose vertices comprise the set Z_d^n, and whose (labelled) arcs are specified by the permutations $\pi(\beta; d)$ of Z_d^n ($\beta \in Z_d$ and n being clear from context) defined as follows.[20] For each $\alpha \in Z_d$ and $x \in Z_d^{n-1}$,

$$(\alpha x)\pi(0; d) = (\alpha x)\pi[0; d] = x\alpha$$

and, for $\beta \neq 0$,

$$(x\alpha)\pi(\beta; d) = x(\alpha + \beta (\text{mod } d));$$

see Fig. 4, where the conventional, base-2, Shuffle-Exchange graph is depicted.

Let $\Pi_d^{\Sigma} =_{\text{def}} \{\pi(\beta; d) \mid \beta \in Z_d\}$.

[19] By Lemma 2, this is the subgroup that yields the structure of the DeBruijn graph.

[20] The permutation $\pi(1; 2)$ is termed an *exchange*.

Lemma 5 *For all d, $Gr(\Pi_d^{\Sigma})$ is isomorphic to the wreath product $Z_d \otimes_{wr} Z_n$.*[21]

Proof. By Lemma 2, it will suffice to prove that for all d, $Gr(\Pi_d^{\Sigma}) = Gr(\Pi_d^{\Delta})$. To this end, note that for all d: $\pi[0;d] = \pi(0;d)$, and for $\beta \in Z_d - \{0\}$,

$$\pi[\beta;d] = \pi(\beta;d)\pi(0;d);$$

and

$$\pi(\beta;d) = \pi[\beta;d](\pi[0;d])^{n-1}$$

(see footnote 16). □

Lemma 6 *For all d, n, the base-d n-dimensional Shuffle-Exchange graph $\Sigma(n;d)$ is isomorphic to the Coset graph*

$$Cos(\mathcal{G};\mathcal{H};\Pi_d^{\Sigma})$$

where $\mathcal{G} = Z_d \otimes_{wr} Z_n$ and $\mathcal{H} = \{0\} \otimes_{wr} Z_n$.

Proof. The proof is virtually identical to that of Lemma 3, so is left to the reader. □

B. Cube-Connected Cycles Networks

Let d, n be a positive integer. The *base-d n-level Cube-Connected Cycles graph* (*CCC graph*, for short) $C(n;d)$ has vertex-set

$$V_{n;d} = Z_n \times Z_d^n.$$

The subset $V_{n;d}^{(\ell)} = \{\ell\} \times Z_d^n$ of $V_{n;d}$ $(0 \le \ell < n)$ is the ℓ^{th} *level* of $C(n;d)$. The edges of $C(n;d)$ are of two varieties. First there are the *inter-level* edges that connect each vertex

$$\langle \ell,\ \beta_0\beta_1\cdots\beta_{n-1}\rangle$$

on level ℓ of $C(n;d)$ $(0 \le \ell < n$; α and each $\beta_i \in Z_d)$ with the corresponding vertex

$$\langle \ell+1(\text{mod } n),\ \beta_0\beta_1\cdots\beta_{n-1}\rangle$$

on level $\ell+1(\text{mod } n)$ of $C(n;d)$. The remaining, *intra-level*, edges form *d-cliques* (or, copies of the complete graph K_d), as follows: On each level $0 \le \ell < n$, each vertex

$$\langle \ell,\ \beta_0\beta_1\cdots\beta_{\ell-1}\alpha\beta_{\ell+1}\cdots\beta_{n-1}\rangle$$

(α and each β_i in Z_d) is adjacent to all vertices

$$\langle \ell,\ \beta_0\beta_1\cdots\beta_{\ell-1}\omega\beta_{\ell+1}\cdots\beta_{n-1}\rangle$$

for all $\omega \in Z_d$.

[21] The set Π_d^{Σ} is often called the *standard* set of generators for $Z_d \otimes_{wr} Z_n$.

There is a natural way to turn $C(n;d)$ into a tragraph whose arcs are labelled with permutations from Π_d^Σ. First, we form the *directed* version $\vec{C}(n;d)$ of $C(n;d)$, by orienting each inter-level edge of $C(n;d)$ from level ℓ to level $\ell+1$ (before reducing $\ell+1$ modulo d); then, we replace each intra-level edge of $C(n;d)$ by mated opposing arcs. Next, we form the *tragraph* version $\tilde{C}(n;d)$ of $\vec{C}(n;d)$, as follows. First, we label each inter-level arc with the permutation $\pi(0;d)$; then, for each $\ell \in Z_n$ and $\alpha, \omega \in Z_d$, we label the arc from vertex

$$\langle \ell,\ \beta_0\beta_1\cdots\beta_{\ell-1}\alpha\beta_{\ell+1}\cdots\beta_{n-1}\rangle$$

to vertex

$$\langle \ell,\ \beta_0\beta_1\cdots\beta_{\ell-1}\omega\beta_{\ell+1}\cdots\beta_{n-1}\rangle$$

with the permutation $\pi(\omega - \alpha(\mathrm{mod}\ d);\ d)$; see Fig. 5.

Lemma 7 *For all d, n, the base-d n-level CCC network $\tilde{C}(n;d)$ is isomorphic (*as a tragraph*) to the Cayley graph $Cay(\Pi_d^\Sigma)$.*

Theorem 4 *For all d, n, the base-d n-dimensional Shuffle-Exchange graph $\Sigma(n;d)$ is a coset graph of the base-d n-level CCC network $\tilde{C}(n;d)$.*

Proof Sketch. The proofs of Lemma 7 and Theorem 4 are almost identical to those of Lemma 1 and Theorem 3, respectively, so are left to the reader; see Fig. 6.

ACKNOWLEDGMENTS: This research was supported in part by NSF Grant DCI-87-96236. It is a pleasure to thank Dave Barrington for helpful advice and technical assistance. Adi Shamir and Don Coppersmith pointed us toward a number of useful group-theoretic results. Comments and suggestions by Tom Leighton and Charles Leiserson helped us improve the presentation.

4. REFERENCES

[**AHU**] A.V. Aho, J.E. Hopcroft, J.D. Ullman (1974): *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, MA.

[**AK1**] S.B. Akers and B. Krishnamurthy (1984): Group graphs as interconnection networks. *14th IEEE International Conference on Fault-tolerant Computing,* 422-427.

[**AK2**] S.B. Akers and B. Krishnamurthy (1986): A group-theoretic model for symmetric interconnection networks. *Intl. Conf. Parallel Processing,* 216-223.

[**AK3**] S.B. Akers and B. Krishnamurthy (1987): On group graphs and their fault tolerance. *IEEE Trans. Comp., C-36,* 885-888.

[AEH | J. Akiyama, H. Era, F. Harary (1983): Regular graph containing a given graph. *Elem. Math. 38,* 15-17.

[BCHLR | S.N. Bhatt, F.R.K. Chung, J.-W. Hong, F.T. Leighton, A.L. Rosenberg (1987): Optimal simulations by Butterfly networks. Typescript, Univ. Massachusetts.

[BM | J.A. Bondy and U.S.R. Murty (1976): *Graph Theory with Applications.* North-Holland, New York.

[CP | A.H. Clifford and G.B. Preston (1967): *The Algebraic Theory of Semigroups, II.* Mathematical Surveys No. 7, American Math. Soc., Providence, RI.

[dB | N.G. DeBruijn (1946): A combinatorial problem. *Proc. Akademe Van Weteschappen 49,* Part 2, 758-764.

[EK1 | P. Erdös and P. Kelly (1963): The minimal regular graph containing a given graph. *American Math. Monthly 70,* 264-274.

[EK2 | P. Erdös and M.S. Krishnamoorthy (1987): On regularizing graphs. Typescript, RPI.

[Fr | H. Fredricksen (1982): A survey of full length nonlinear shift register cycle algorithms. *SIAM Review 24,* 195-221.

[Ha | M. Hall, Jr. (1959): *The Theory of Groups.* Macmillan Co.

[ISO | M. Imase, T. Soneoka, K. Okada (1985): Connectivity of regular directed graphs with small diameters. *IEEE Trans. Comp., C-34,* 267-273.

[Ko | A. Kotzig (1969): The decomposition of a directed graph into quadratic factors consisting of cycles. *Acta FRN Univ. Comment. Math., XXII,* 27-29.

[PV | F.P. Preparata and J.E. Vuillemin (1981): The cube-connected cycles: a versatile graph for parallel computation. *C. ACM 24,* 300-309.

[Ro1 | A.L. Rosenberg (1971): Data graphs and addressing schemes. *J. CSS 5,* 193-238.

[Ro2 | A.L. Rosenberg (1974): An extrinsic characterization of addressable data graphs. *Discrete Math. 9,* 61-70.

[Ul | J.D. Ullman (1984): *Computational Aspects of VLSI.* Computer Science Press, Rockville, MD.

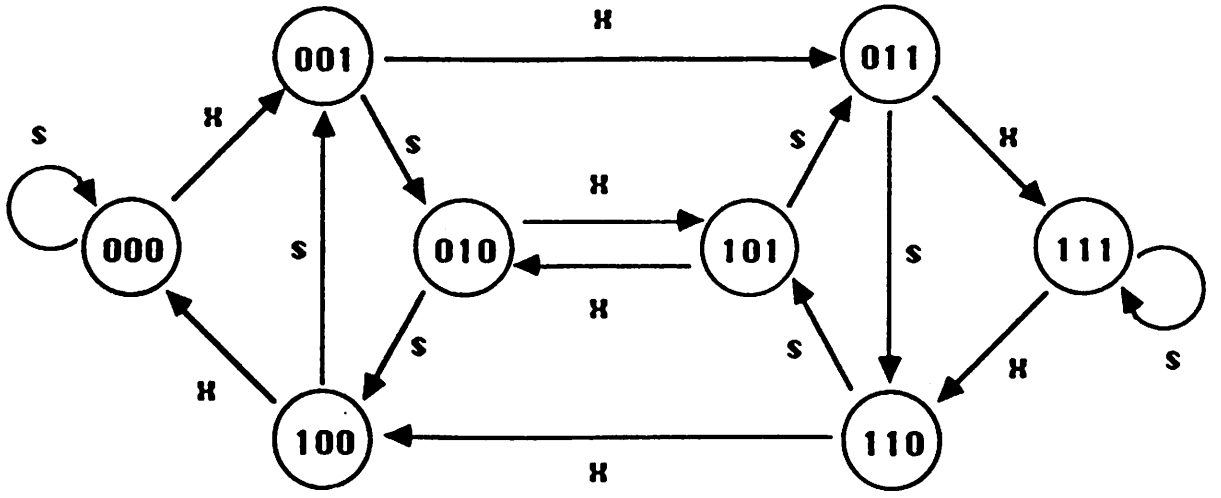


Figure 1. The DeBruijn graph $\Delta(3;2)$ as a GAG.

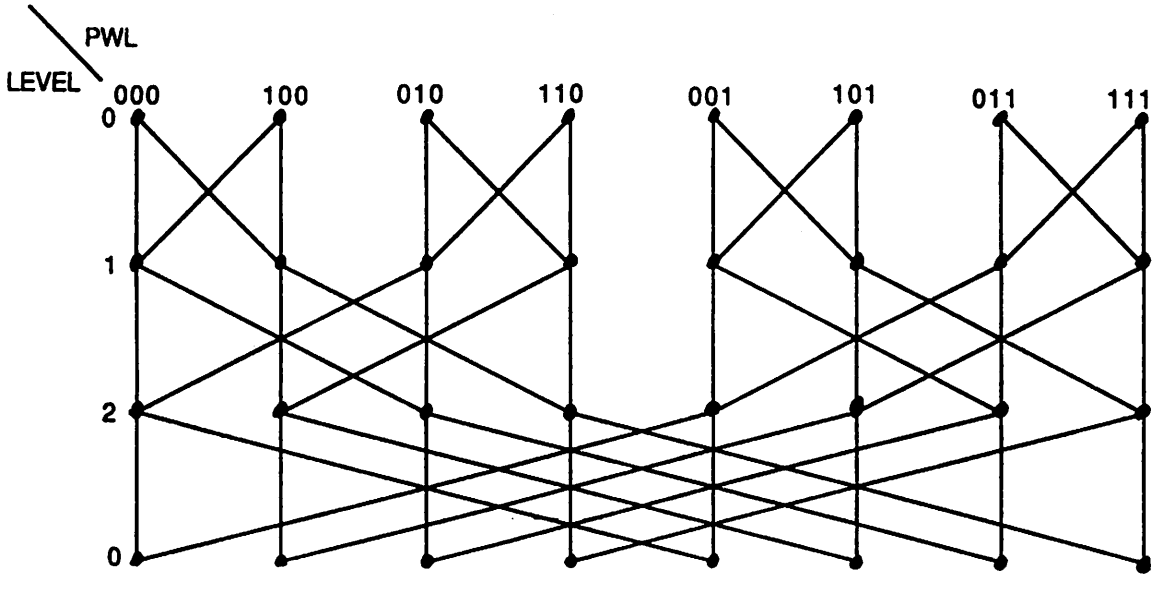
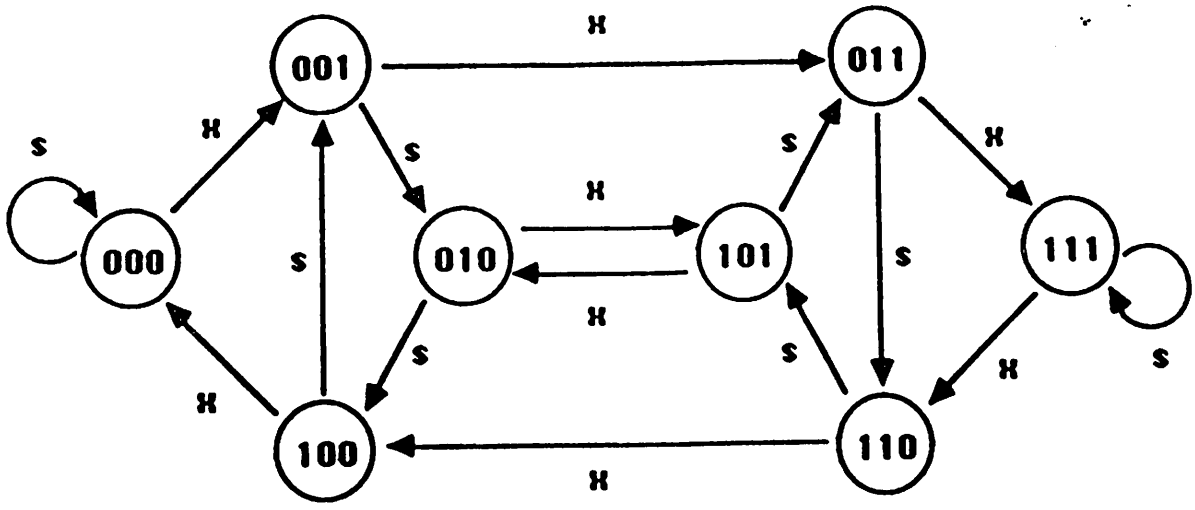


Figure 2. The Butterfly network $\tilde{B}(3;2)$.



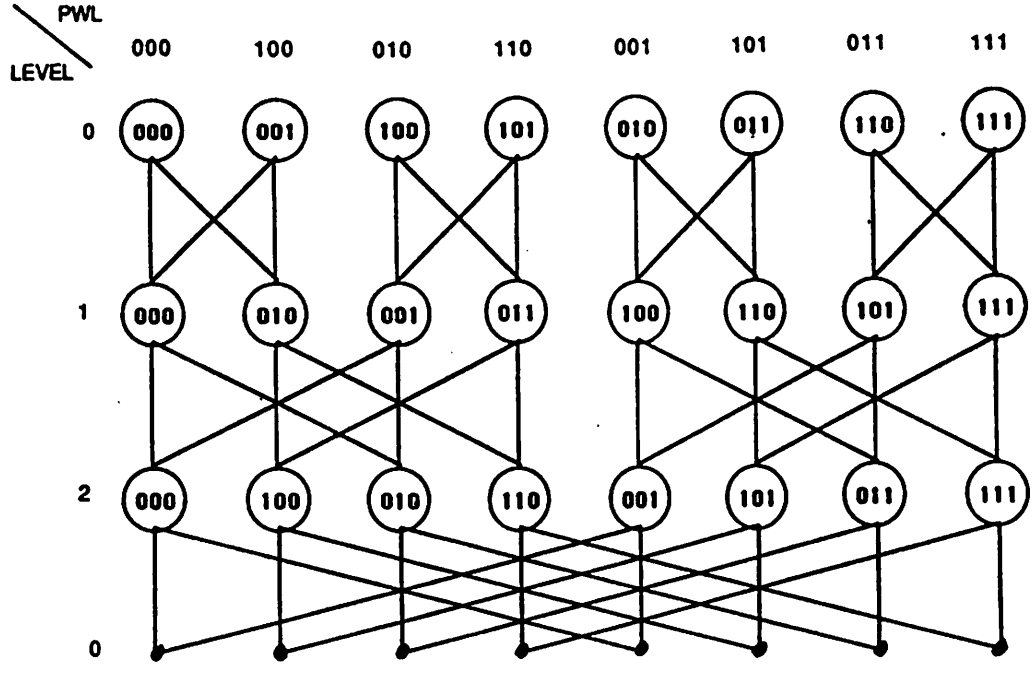


Figure 3. Illustrating $\Delta(3;2)$ as a coset graph of $\tilde{B}(3;2)$.

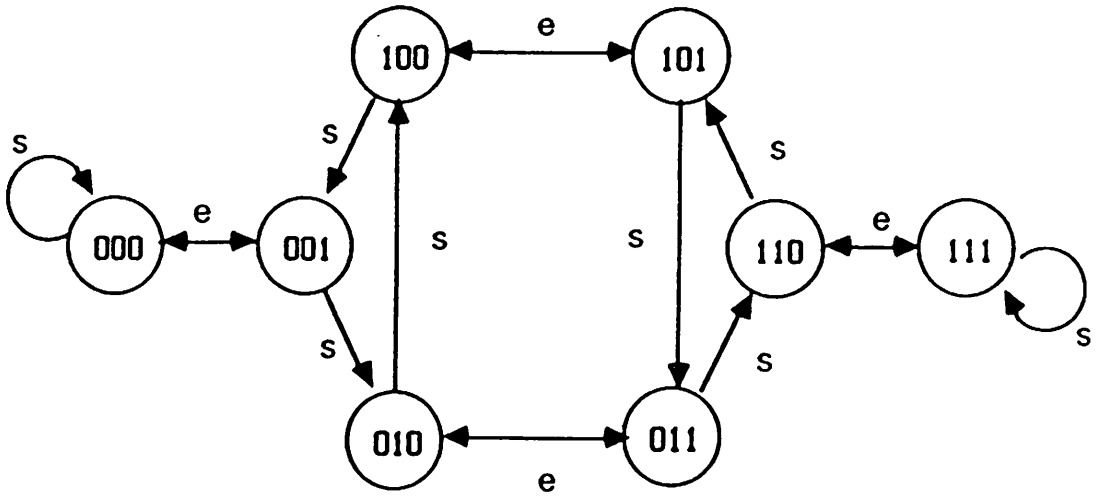


Figure 4. The Shuffle-Exchange graph $\Sigma(3;2)$ as GAG.

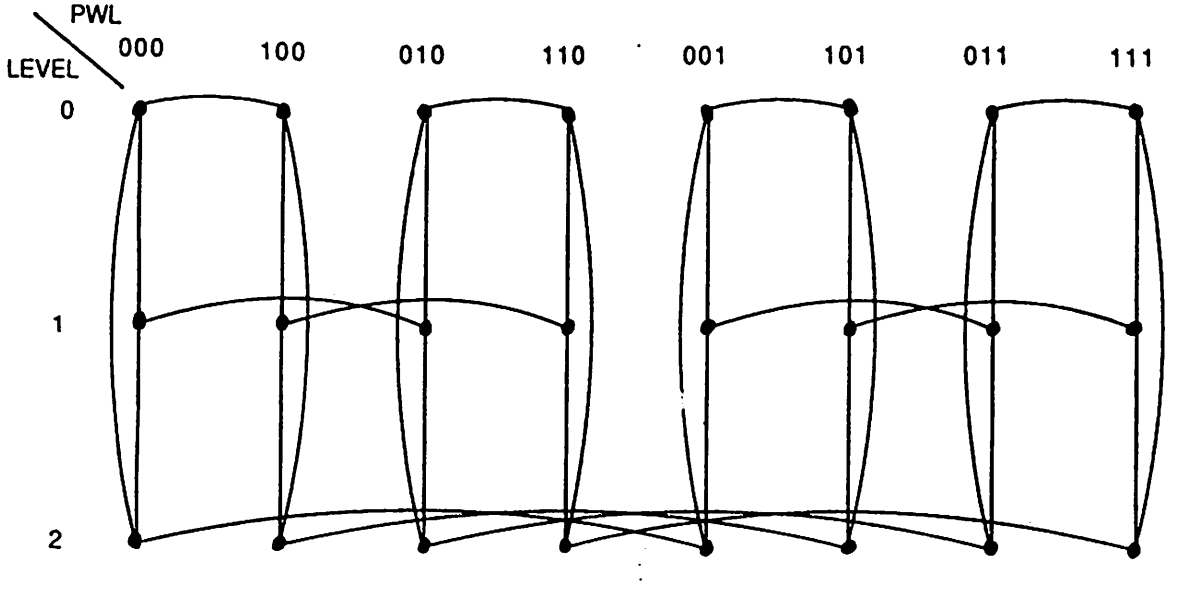
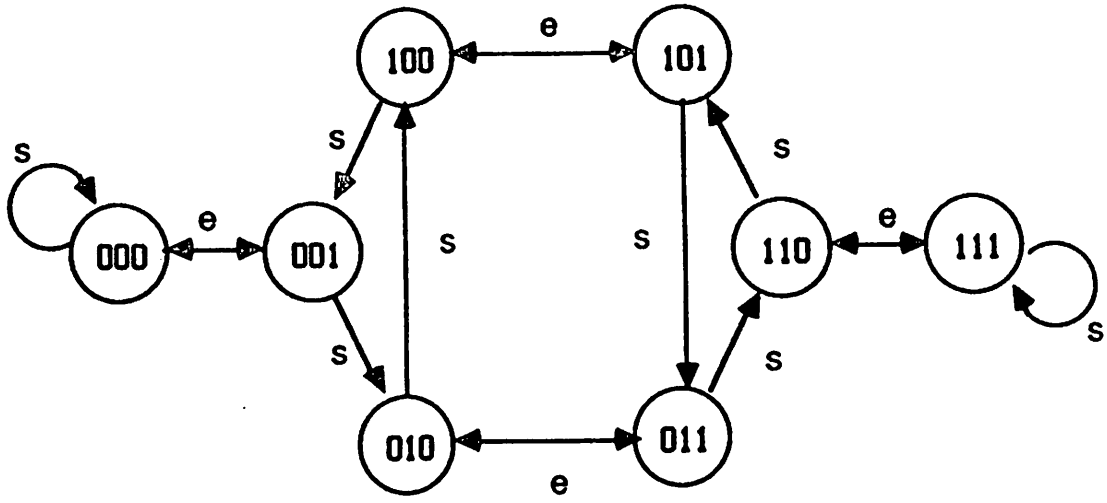


Figure 5. The Cube-Connected Cycles network $\hat{C}(3;2)$.



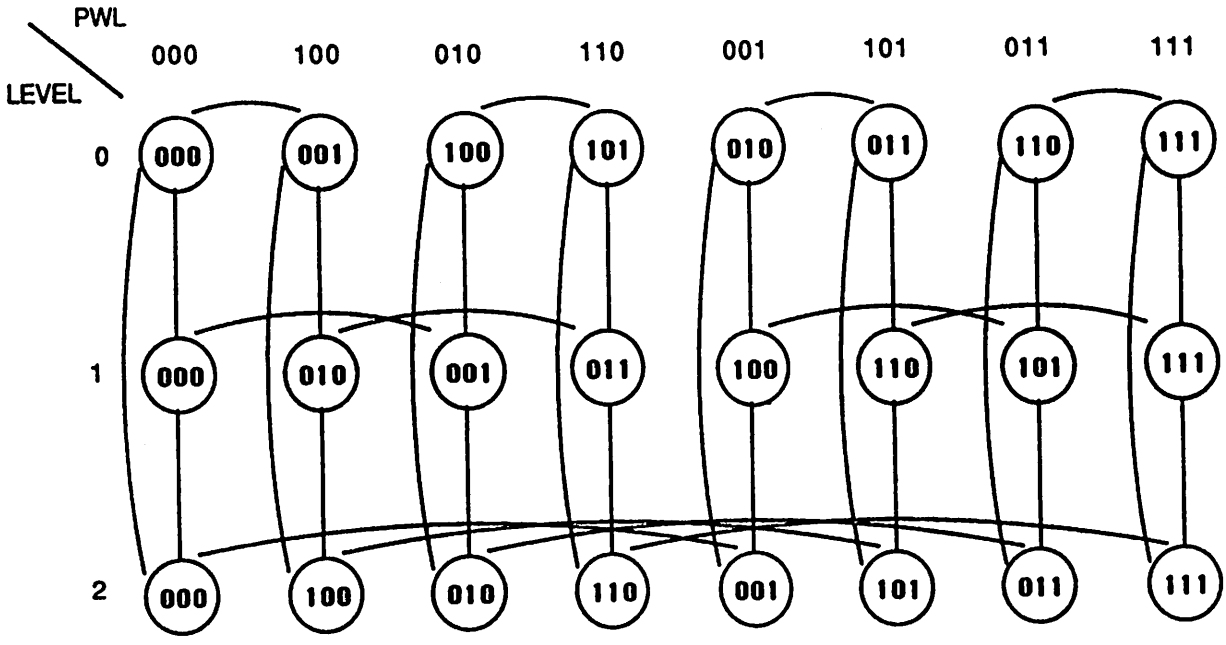


Figure 6. Illustrating $\Sigma(3;2)$ as a coset graph of $\tilde{C}(3;2)$.