Information Processing Letters 22 (1986) 285–289
North-Holland

30 May 1986

ON RECURRENT AND RECURSIVE INTERCONNECTION PATTERNS

Ian PARBERRY

Department of Computer Science, College of Science, The Pennsylvania State University, 333 Whitmore Laboratory, University Park, PA 16802, U.S.A.

Communicated by David Gries
Received 5 April 1985
Revised 12 August 1985

A number of trivalent graphs, in particular variants of the cube-connected cycles and shuffle-exchange, have become popular as interconnection patterns for synchronous parallel computers. We consider highly-structured interconnection patterns that allow large parallel machines to be constructed from isomorphic copies of smaller ones, plus (perhaps) a few extra processors. If only a small number of extra processors are added, we call the interconnection pattern *recurrent*. If no extra processors are added, we call it *recursive*. We show that a constant-degree recursive interconnection pattern is, in a sense, not as versatile as the cube-connected cycles or shuffle-exchange, and we present a trivalent recurrent interconnection pattern that is.

Keywords: Parallel computation, interconnection pattern, composite algorithm, cube-connected cycles

1. Introduction

An *interconnection pattern* is an infinite series $G = (G_0, G_1, G_2, \ldots)$ of finite graphs. Graph G_n represents a parallel machine, each vertex a processor, and each edge a communication link between processors. The processor bound P(n) is the number of vertices in G_n as a function of n. For an interconnection pattern to be of any practical use, the following properties must hold:

(1) The degree of G_n is constant (i.e., independent of n).

(2) G_n is easy to compute (as a function of n).

(3) There is a constant $c > 0$ such that, for all $n \geqslant 1$, $P(n) \leqslant c\,P(n-1)$.

The literature already provides us with useful interconnection patterns. Preparata and Vuillemin [5] studied a useful class of algorithms (which they call *composite* algorithms) for the multi-dimensional cube. Although this interconnection pattern has nonconstant degree, they presented a practical interconnection pattern, called the *cube-connected cycles*, which has the ability to simulate composite algorithms without asymptotic time loss. We call a practical interconnection pattern with this property *composite*. The shuffle-exchange interconnection pattern [7] is also easily seen to be composite. There are efficient composite algorithms for many useful data routing problems (such as sorting and performing permutations), which can thus efficiently be implemented on either the cube-connected cycles or shuffle-exchange.

Loosely speaking, an interconnection pattern is said to be *recurrent* if each graph G_n is made up of many isomorphic copies of smaller graphs G_m where $m < n$. Both the cube-connected cycles and shuffle-exchange are composite, but neither is recurrent. Meyer auf der Heide [2,3] has given a degree-4 interconnection pattern that is both composite and recurrent. We present a degree-3 interconnection pattern with the same properties. Each G_n is made up of at least $P(n)/(2P(m))$ copies of G_m. Also, we find that it is impossible to design a composite interconnection pattern with the property that each G_n is made up of exactly $P(n)/P(m)$ copies of G_m. We call an interconnection pattern with the latter property *recursive*.

The main body of this paper is divided into two sections. In Section 2 we demonstrate that no recursive interconnection pattern can permute n items in O(log n) time, a task that is well within the abilities of a composite interconnection pattern. In Section 3 we give a composite recurrent interconnection pattern, which we call the *cube-connected lines*. A preliminary version of the results of this paper has appeared in [4].

2. Recursive interconnection patterns

An interconnection pattern $G = (G_0, G_1, \ldots)$ with P(n) processors is said to be *recurrent* if, for all n, m with $0 \leqslant m \leqslant n$, G_n has $\Omega(P(n)/(P(m))$ disjoint subgraphs isomorphic to G_m. The simplest form of recurrence one might choose is to have G_n constructed from *precisely* P(n)/P(m) such subgraphs. Unfortunately, this type of recurrent interconnection pattern is much less powerful than the shuffle-exchange [7] or cube-connected cycles [5] interconnection patterns.

Suppose c is a fixed positive integer (independent of n). More precisely, a *recursive* interconnection pattern is one in which G_n $(n > 0)$ is made up of exactly c disjoint copies of G_{n-1} (with some fixed graph for G_0), joined by extra edges from some graph G'_n.

Theorem 2.1. *A constant degree recursive parallel machine with* P(n) *processors cannot permute* P(n) *items in* O(log P(n)) *steps.*

Proof. For a contradiction, suppose $G = (G_0, G_1, \ldots)$ is a P(n)-processor, degree-d recursive interconnection pattern that can be used to permute P(n) items in O(log P(n)) time. The following simple and elegant technique is due to Meertens [1].

Without loss of generality assume $P(0) = 1$ (note that this means $P(n) = c^n$). For convenience, write P_n for P(n). Let E_n denote the number of edges in G_n, E'_n denote the number of edges in G'_n, and $\Gamma_n = E_n/P_n$. Note that $\Gamma_n \leqslant \frac{1}{2}d$. (Let S_n be the sum over all vertices v in G_n of the number of edges incident with v. Clearly, $S_n \leqslant d\, P_n$. But every edge is counted twice, so $S_n = 2E_n$.)

We claim that, for $n \geqslant 1$, $E'_n = \Omega(c^n/n)$. Consider one of the subgraphs of G_n isomorphic to G_{n-1}. Pick a permutation that takes a data item from each vertex of the subgraph (there are c^{n-1} of them) to a vertex of G_n outside that subgraph. These data items must pass along the edges of G'_n, since these are the only edges linking the subgraph with the rest of G_n. Thus, in one step at most E'_n items can be moved. By hypothesis we can move all the items in O(n) steps. There are c^{n-1} items to be moved. Hence, $c^n = O(E'_n n)$. This is sufficient to prove the above claim.

Therefore,

$$
\begin{aligned}
E_n &= E'_n + c\, E_{n-1} \\
&= \sum_{i=1}^{n-1} c^i E'_{n-i} \\
&= \Omega\left(\sum_{i=1}^{n-1} c^i c^{n-i}/(n-i) \right) \quad \text{(by the claim)} \\
&= \Omega\left(\sum_{i=1}^{n-1} c^n/i \right) \quad \text{(by re-indexing).}
\end{aligned}
$$

Thus,

$$\Gamma_n = E_n/P_n = \Omega\left(\sum_{i=1}^{n-1} 1/i \right),$$

which diverges as $n \to \infty$. But this contradicts the fact that $\Gamma_n \leqslant \frac{1}{2}d$, a constant independent of n. Thus, no such parallel machine can exist. □

This is in contrast to the corresponding result for the cube-connected cycles (see [5]) and shuffle-exchange (see [4]).

3. A recurrent interconnection pattern

First, let us introduce some useful notation. Suppose v and i are nonnegative integers. If $i \geqslant 1$, then let v_i denote the *i*th least-significant bit in the binary representation of v, that is, $v_i = \lfloor v/2^{(i-1)} \rfloor \bmod 2$. Where convenient, we may confuse the integer v and a binary representation $v_k v_{k-1} \ldots v_1$ (where $k \geqslant \lfloor \log v \rfloor + 1$) of v. Also, let $v^{(i)}$ denote the integer that differs from v precisely in the i*th* (least-significant) bit, that is, $v^{(i)} = v + (-1)^{v_i} 2(i-1)$.

The *cube-connected cycles* CCC_k of Preparata and Vuillemin [5] is defined as follows. Let r be such that $2^{r-1} + r - 1 < k \leqslant 2^r + r$. CCC_k has vertex-set

$$\{(v, p) \mid 0 \leqslant v < 2^{k-r}, 0 \leqslant p < 2^r\},$$

and each vertex (v, p) is joined to the following vertices:

(i) $(v^{(p+1)}, p)$, provided $0 \leqslant p < k - r$,

(ii) $(v, (p + 1) \bmod 2^r)$, and

(iii) $(v, (p - 1) \bmod 2^r)$.

The first link is called a *cube* edge, the remaining two, *cycle* edges. CCC_k has 2^k vertices and has degree 3.

The following is a recurrent interconnection pattern that is as powerful as the cube-connected cycles, at least in its ability to simulate composite algorithms. The *cube-connected lines*, CCL_k (see Fig. 1) is simply a copy of CCC_k with the edges from vertices (v, 0) to $(v, 2^r - 1)$, $0 \leqslant v < 2^{k-r}$ deleted (we call the remaining cycle edges *line edges*, and the deleted cycle edges *external edges*). That is, the cycles of the cube-connected cycles are broken, and thus become lines. CCL_k has 2^k vertices and has degree 3.

It is fairly easy to see that CCL_k is recurrent. We need to differentiate the special case of CCL_k when k is of the form $2^r + r$, for some r. In this case we call CCL_k a *full* cube-connected lines graph.

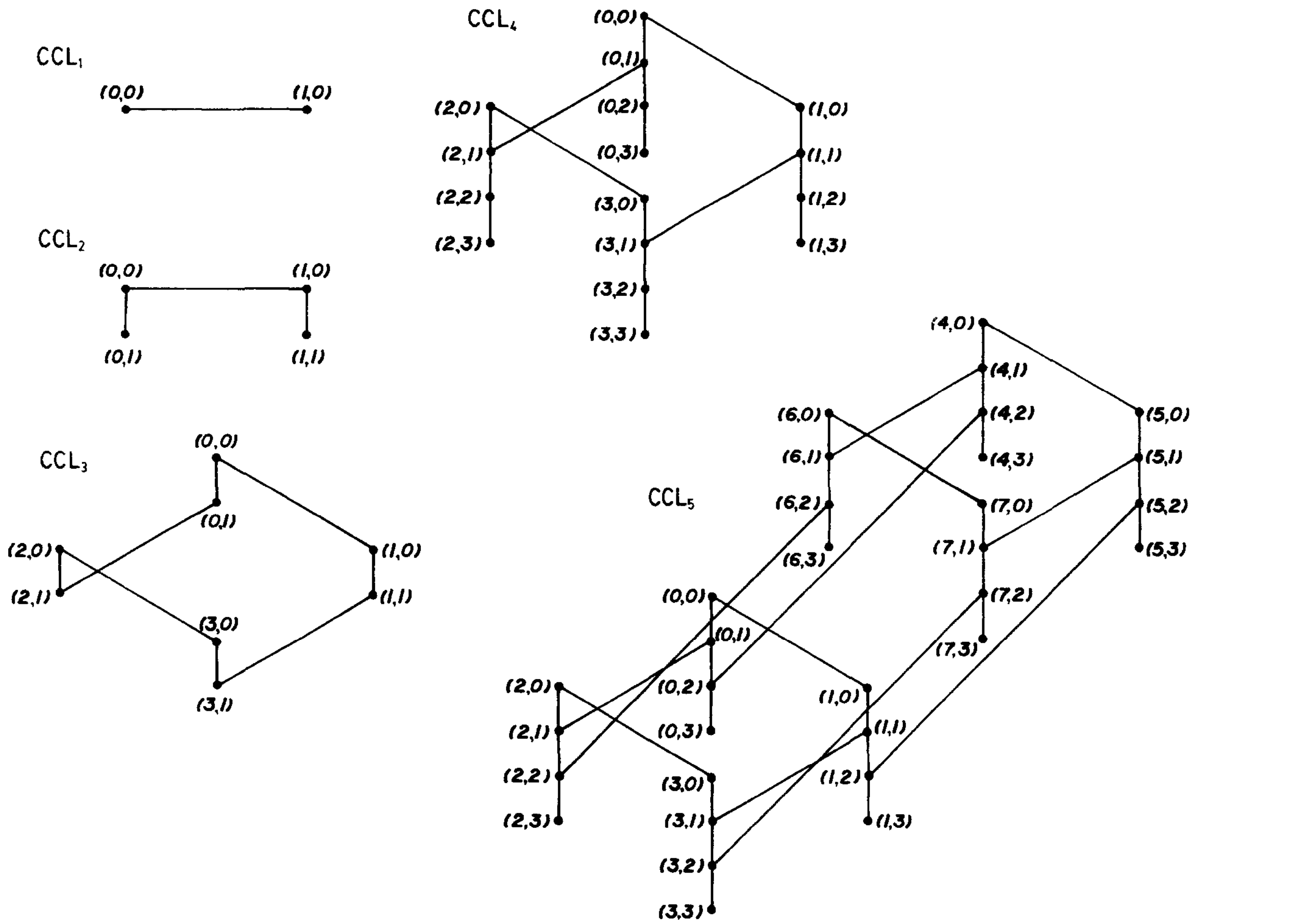

Fig. 1. The 2, 4, 8, 16, and 32 vertex cube-connected lines graphs, CCL_1 through CCL_5. Line-edges are drawn vertically; the remainder are cube-edges.

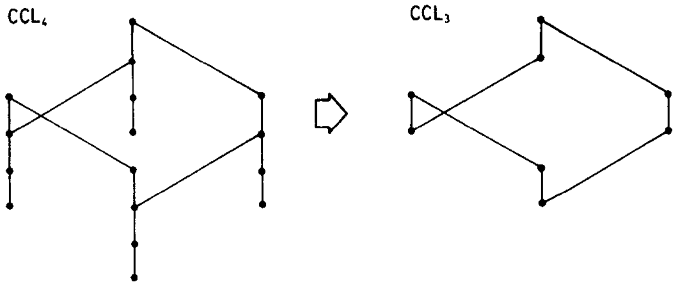

Fig. 2. CCL_4 has one subgraph isomorphic to CCL_3.

Lemma 3.1. *If* $k = 2^r + r$, *then* CCL_{k+1} *has exactly one subgraph isomorphic to* CCL_k.

Proof. Suppose $k = 2^r + r$. CCL_k has vertices (v, p) with $0 \leqslant v < 2^{k-r}$, $0 \leqslant p < 2^r$. Vertex (v, p) is joined to the following vertices:

(i) $(v^{(p+1)}, p)$, $0 \leqslant v < 2^{k-r}$, $0 \leqslant p < 2^r$,
(ii) $(v, p+1)$, $0 \leqslant v < 2^{k-r}$, $0 \leqslant p < 2^r - 1$, and
(iii) $(v, p-1)$, $0 \leqslant v < 2^{k-r}$, $0 < p < 2^r$.

CCL_{k+1} has vertices (v, p) with $0 \leqslant v < 2^{k-r}$, $0 \leqslant p < 2^{r+1}$. Vertex (v, p) is joined to the following vertices:

(i) $(v^{(p+1)}, p)$, $0 \leqslant v < 2^{k-r}$, $0 \leqslant p < 2^r$,
(ii) $(v, p+1)$, $0 \leqslant v < 2^{k-r}$, $0 \leqslant p < 2^{r+1} - 1$, and
(iii) $(v, p-1)$, $0 \leqslant v < 2^{k-r}$, $0 < p < 2^{r+1}$.

Thus, CCL_k looks exactly like CCL_{k-1} with lines extended to double the length using vertices without cube links (see Fig. 2). So, CCL_{k+1} has exactly one subgraph isomorphic to CCL_k. □

Lemma 3.2. *If* k *is not of the form* $2^r + r$, *then* CCL_{k+1} *has two disjoint subgraphs isomorphic to* CCL_k.

Proof. Without loss of generality, suppose $k < 2^r + r$. CCL_k has vertices (v, p) with $0 \leqslant v < 2^{k-r}$, $0 \leqslant p < 2^r$. Vertex (v, p) is joined to the following vertices:

(i) $(v^{(p+1)}, p)$, $0 \leqslant v < 2^{k-r}$, $0 \leqslant p < k - r$,
(ii) $(v, p+1)$, $0 \leqslant v < 2^{k-r}$, $0 \leqslant p < 2^r - 1$, and
(iii) $(v, p-1)$, $0 \leqslant v < 2^{k-r}$, $0 < p < 2^r$.

CCL_{k+1} has vertices (v, p) with $0 \leqslant v < 2^{k-r+1}$, $0 \leqslant p < 2^r$. Vertex (v, p) is joined to the following vertices:

(i) $(v^{(p+1)}, p)$, $0 \leqslant v < 2^{k-r+1}$, $0 \leqslant p < k - r + 1$,
(ii) $(v, p+1)$, $0 \leqslant v < 2^{k-r+1}$, $0 \leqslant p < 2^r - 1$, and
(iii) $(v, p-1)$, $0 \leqslant v < 2^{k-r+1}$, $0 < p < 2^r$.

Thus, deleting the cube-edges from (v, p) to $(v^{(p+1)}, p)$ with $p = k - r$ from CCL_{k+1} gives two disjoint graphs isomorphic to CCL_k (see Fig. 3). □

Lemma 3.3. *If* $k = 2^r + r$ *and* $j = 2^s + s$, *where* $r \geqslant s$, *then* CCL_k *has exactly* 2^{k-j} *disjoint subgraphs isomorphic to* CCL_j.

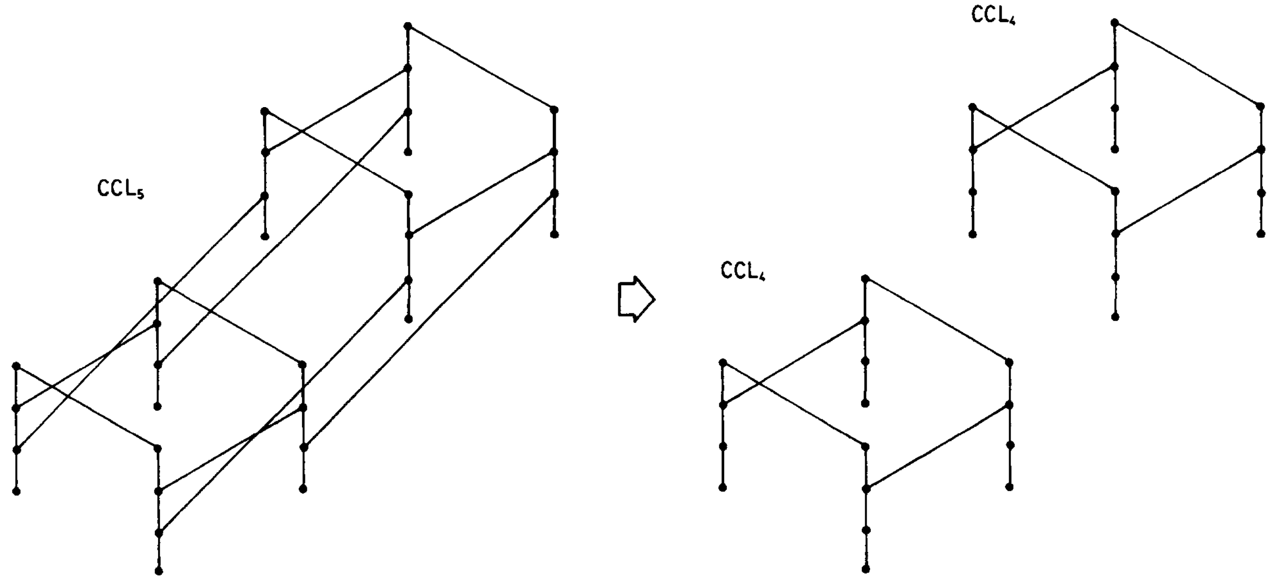

Fig. 3. CCL_5 has two subgraphs isomorphic to CCL_4.

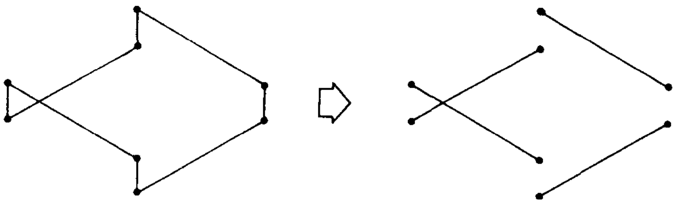

Fig. 4. CCL_3 has four subgraphs isomorphic to CCL_1.

Proof. Suppose $k = 2^r + r$ and $j = 2^s + s$ for some $r \geqslant s \geqslant 0$. CCL_j has vertices (v, p), $0 \leqslant v < 2^{2^s}$, $0 \leqslant p < 2^s$. Vertex (v, p) is joined to the following vertices:

(i) $(v^{(p+1)}, p)$, $0 \leqslant v < 2^{2^s}$, $0 \leqslant p < 2^s$,

(ii) $(v, p+1)$, $0 \leqslant v < 2^{2^s}$, $0 \leqslant p < 2^s - 1$, and

(iii) $(v, p-1)$, $0 \leqslant v < 2^{2^s}$, $0 < p < 2^s$.

CCL_k has vertices (v, p), $0 \leqslant v < 2^{2^r}$, $0 \leqslant p < 2^r$. Vertex (v, p) is joined to the following vertices:

(i) $(v^{(p+1)}, p)$, $0 \leqslant v < 2^{2^r}$, $0 \leqslant p < 2^r$,

(ii) $(v, p+1)$, $0 \leqslant v < 2^{2^r}$, $0 \leqslant p < 2^r - 1$, and

(iii) $(v, p-1)$, $0 \leqslant v < 2^{2^r}$, $0 < p < 2^r$.

Deleting the line-edges between vertices $(v, i\,2^s - 1)$ and $(v, i\,2^s)$ for $0 \leqslant v < 2^{2^r}$, $0 \leqslant i < 2^{r-s}$, breaks CCL_k into 2^{k-j} graphs isomorphic to CCL_j (see Fig. 4). Thus, a full CCL_k has 2^{k-j} disjoint subgraphs isomorphic to a full CCL_j. □

Theorem 3.4. *For* $0 \leqslant j \leqslant k$, CCL_k *has at least* 2^{k-j-1} *disjoint subgraphs isomorphic to* CCL_j.

Proof (*Sketch*). The result easily follows using the above lemmas. First, reduce CCL_k into subgraphs isomorphic to the next smaller full CCL, using Lemmas 3.1 and 3.2. If CCL_j is encountered along the way, then this is sufficient. Next, using Lemma 3.3, reduce the full CCL immediately below CCL_k into subgraphs isomorphic to the full CCL immediately above CCL_j. The latter can be reduced to CCL_j by application of Lemma 3.2.

In this entire process we only once have to reduce a nonfull CCL to subgraphs isomorphic to full ones. Thus, CCL_k consists of 2^{k-j-1} subgraphs isomorphic to CCL_j. □

Note that any attempt to increase the number of subgraphs from 2^{k-j-1} to 2^{k-j} is doomed to failure. For if CCL_k had 2^{k-j} subgraphs isomorphic to CCL_j, it would then be recursive. Thus, by Theorem 2.1 it would be much weaker than the cube-connected cycles for computing permutations. However, we have the following theorem.

Theorem 3.5. *A cube-connected lines with* 2^k *processors can simulate a* 2^k *processor composite algorithm without asymptotic time loss.*

Proof. The proof is almost identical to that for the cube-connected cycles [5]. In that proof:

(1) The pipelining phase utilizes a synchronous cyclic shift around the cycles. This can be replaced with a linear shift along the corresponding lines of the cube-connected lines graph, with wrap-around at the ends (at most doubling the time requirement).

(2) Communication within the cycles is performed using a procedure called LOOPOPER. A close examination of this procedure reveals that it never uses external edges, and thus can be executed on the cube-connected lines graph. □

Thus, in particular, a parallel machine based on the cube-connected lines interconnection pattern can permute n items in O(log n) time.

Reif and Valiant [6] have independently discovered a graph that is similar to the cube-connected lines.

References

[1] L. Meertens, Recurrent ultracomputers are not (log n)-fast, Tech. Rept. IW118/79, Dept. of Computer Science, Centre for Mathematics and Computer Science, 1979.

[2] F. Meyer auf der Heide, Efficiency of universal parallel computers, Acta Inform. 19 (1983) 269–296.

[3] F. Meyer auf der Heide, Infinite cube-connected cycles, Inform. Process. Lett. 16 (1983) 1–2.

[4] I. Parberry, A complexity theory of parallel computation, Ph. D. Thesis, Dept. of Computer Science, Univ. of Warwick, 1984.

[5] F.P. Preparata and J. Vuillemin, The cube-connected cycles: A versatile network for parallel computation, Comm. ACM 24 (5) (1981) 300–309.

[6] J. Reif and L. Valiant, A logarithmic time sort for linear size networks, Proc. 15th Ann. ACM Symp. on Theory of Computing, Boston, MA (1983) 10–16.

[7] H.S. Stone, Parallel processing with the perfect shuffle, IEEE Trans. Comput. C-20 (2) (1971) 153–161.