1-Fair Alternator Designs for the de Bruijn Network

Hsu-Shen Lin, Chang-Biau Yang† and Kuo-Tsung Tseng
Department of Computer Science and Engineering
National Sun Yat-sen University, Kaohsiung, Taiwan
†cbyang@par.cse.nsysu.edu.tw

Abstract

In a 1-fair alternator of a network of concurrent processors, no processor executes the critical step twice when one or more other processors have not executed the critical step yet. In this paper, two algorithms are proposed to solve the coloring (1-fair alternator design) problem on the de Bruijn network. The first one uses $2\lceil \log_2 k \rceil + 1$ *colors to color the* k*-ary de Bruijn graph with two digits, while the second one uses* $p+1$ *only colors, where* $\binom{p-1}{\lfloor (p-1)/2 \rfloor} < k \leq \binom{p}{\lfloor p/2 \rfloor}$*. The second coloring method is optimal when* $k = \binom{p}{\lfloor p/2 \rfloor}$*. Furthermore, the extension of our coloring method can be applied to the* k*-ary de Bruijn graph with three or more digits.*

1. Introduction

A *de Bruijn graph* [1, 2, 3, 4, 12, 11, 8, 9] $dB(k,m)$ consists of k^m nodes, where each node is labeled by an m-vector in the k-ary number system. Node $v = v_m v_{m-1} \cdots v_1$ in a directed de Bruijn graph connects to node $v_{m-1} v_{m-2} \cdots v_1 c$, where c is an arbitrary k-ary digit, $0 \leq c \leq k-1$. If the de Bruijn graph is an undirected one, then node v would also connects to nodes $c v_m v_{m-1} \ldots v_2$. In this paper, we shall focus on undirected de Bruijn networks only. Figure 1 shows the undirected de Bruijn graph $dB(3,2)$, where self-loops are removed and parallel edges are merged.

The routing and broadcasting algorithms on the de Bruijn network have been extensively studied [2, 10]. Mao and Yang [8] first proposed a shortest path routing algorithm for the binary de Bruijn network. They also presented a fault tolerant routing method that two node disjoint paths are provided, one of them is the shortest one and the other one is of length at most $m + \log_2 m + 4$.

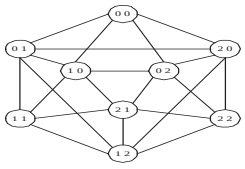


Figure 1. The undirected $dB(3,2)$ graph.

A network of concurrent processors is called an *alternator*, proposed by Gouda and Haddix [5], for a self-stabilizing system, if the following two conditions hold. (1) If one processor executes the critical step, then none of its neighbors executes the critical step at the same time. (2) Each processor executes the critical step infinitely often along any concurrent execution. And it is said to be *1-fair* if condition (2) is changed as: A processor can execute the critical step again only if all other processors have executed the critical step once.

Condition (1) may be taken as the exclusive property of the coloring problem. In other words, the set of nodes with the same color can execute the critical step concurrently since they are not adjacent. The performance of a 1-fair alternator design depends on how often each processor can execute the critical step, i.e. the number of colors used.

For alternator design, Gouda and Haddix [5] proposed a simple method for a multiprocessor system. Huang and Chen [6] proposed an approach for designing optimal 1-fair alternators for hypercubes and for $D \times D$ mesh with odd D. Mao and Yang [9] proposed a 1-fair alternator design for the binary de Bruijn graph $dB(2,m)$. It is optimal and each processor executes the critical step in every three steps. Mao [7] also provided a design for $dB(k,m)$ which allows each processor executes the critical steps in every $k+1$ steps. In other words, the number of colors used to color $dB(k,m)$ is $k+1$.

In this paper, two algorithms are proposed to solve the coloring problem on de Bruijn graphs. The rest of this paper is organized as follows. In Section 2, we shall propose an algorithm which uses $2\lceil \log_2 k \rceil + 1$ colors to color $dB(k,2)$. In Section 3, an improved algorithm uses only $p+1$ colors to color $dB(k,2)$, where $\binom{p-1}{\lfloor (p-1)/2 \rfloor} < k \leq \binom{p}{\lfloor p/2 \rfloor}$. In Section 4, our coloring method are extended from $dB(k,2)$ to $dB(k,m)$ for any $m \geq 3$ without requiring more colors.

Finally, we give some conclusions in Section 5.

2. A Coloring Method with $2\log_2 k + 1$ Colors

We shall apply the divide-and-conquer strategy to solve the problem. First, we partition the k numbers $\{0, 1, \cdots, k-1\}$ into two disjoint sets S_0 and S_1 arbitrarily. Then the nodes in $dB(k, 2)$ can be divided into four groups, which are labeled as 00, 01, 10, and 11, and each of them can be viewed as a *supernode* in $dB(2, 2)$. The node $v = v_2v_1$ belongs to the supernode ij if $v_2 \in S_i$ and $v_1 \in S_j$, where $i, j \in \{0, 1\}$. Take $dB(4, 2)$ for example, suppose we divide $\{0, 1, 2, 3\}$ into two disjoint sets $S_0 = \{0, 1\}$ and $S_1\{2, 3\}$ arbitrarily. Then, the nodes in each supernode are listed as follows:

$$Supernode\ 00 = \{00, 01, 10, 11\}$$
$$Supernode\ 01 = \{02, 03, 12, 13\}$$
$$Supernode\ 10 = \{20, 21, 30, 31\}$$
$$Supernode\ 11 = \{22, 23, 32, 33\}$$

Figure 2 shows the relationship between $dB(4, 2)$ and $dB(2, 2)$. A node in $dB(2, 2)$ has a self-loop if and only if edges exist within the corresponding supernode in $dB(k, 2)$. Thus, if a node in $dB(2, 2)$ has no self-loop, then the corresponding supernode in $dB(k, 2)$ forms an independent set.

It is clear that both supernodes 01 and 10 are independent sets and they can be colored with two colors, one for each supernode. Supernodes 00 and 11 are nodes with self-loop in $dB(2, 2)$, so they are not independent sets and none of them can be colored with one color. However, these can be viewed as subproblems, to which the same algorithm can be applied recursively. Note that supernodes 00 and 11 are two disjoint subgraphs and a color used in one graph can be used in the other one as well. Finally, when we reach the end of the recursion, each supernode would contain only one node, and all these supernodes can be colored with the same color. Our coloring algorithm for $dB(k, 2)$ is given as follows.

Algorithm 1. $2\lceil\log_2 k\rceil + 1$ *Coloring Algorithm*
Input: The de Bruijn graph $dB(k, 2)$.

Output: A valid coloring on $dB(k, 2)$ where $2\lceil\log_2 k\rceil + 1$ colors are used.

Step 1: Divide $\{0, 1, \cdots, k-1\}$ into $S_0 = \{0, 1, \cdots, \lceil\frac{k}{2}\rceil - 1\}$ and $S_1 = \{\lceil\frac{k}{2}\rceil, \lceil\frac{k}{2}\rceil + 1, \cdots, k-1\}$.

Step 2: Assign one color A to all nodes in supernode 01. Assign another color B to all nodes in supernode 10. These two colors must not be used before.

Step 3: Renumber the elements in S_1 to $\{0, 1, \cdots, k - \lceil\frac{k}{2}\rceil - 1\}$. Accordingly, the node labels in supernode 11 are changed. Then solve the coloring problem on supernode 00 and supernode 11 independently and recursively. Note that the colors used in supernodes 00 and 11 could be the same, but cannot be colors A and B.

Table 1. The divide-and-conquer coloring method for $dB(8, 2)$.

	0	1	2	3	4	5	6	7
0	G	E	C	C	A	A	A	A
1	F	G	C	C	A	A	A	A
2	D	D	G	E	A	A	A	A
3	D	D	F	G	A	A	A	A
4	B	B	B	B	G	E	C	C
5	B	B	B	B	F	G	C	C
6	B	B	B	B	D	D	G	E
7	B	B	B	B	D	D	F	G

Table 2. The dividing information in Table 1.

	0	1	2	3	4	5	6	7
A, B	0	0	0	0	1	1	1	1
C, D	0	0	1	1	0	0	1	1
E, F	0	1	0	1	0	1	0	1

As an example, Table 1 shows how this algorithm solves the coloring problem on $dB(8, 2)$ recursively. A node v_2v_1 represented by the intersection of row v_2 and column v_1, and the symbol on the intersection is the color of that node. For example, nodes 25 and 23 are colored with colors A and E, respectively. When we reach the end of the recursion, each supernode corresponds to one node on the diagonal line. As we can see that these nodes on the diagonal line are colored with the same color.

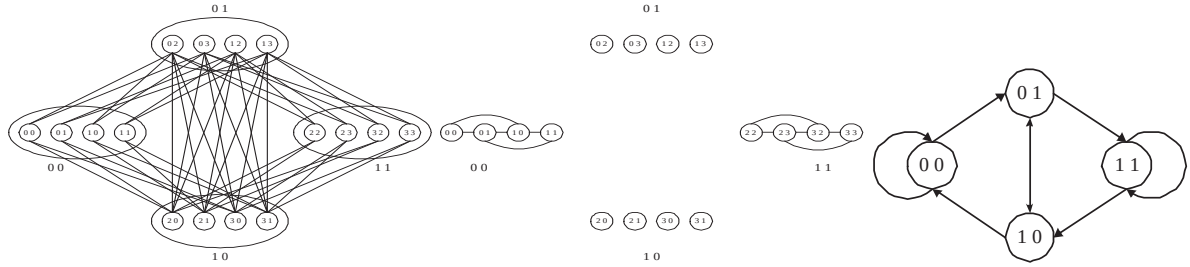
Theorem 1. *Algorithm 1 uses* $2\lceil\log_2 k\rceil + 1$ *colors to color de Bruijn graph* $dB(k, 2)$.

All proofs in this paper are omitted because of the page number limit.

3. An Improved Coloring Method

In the previous section, we divide k into two disjoint sets. Actually, if we keep the result how we divide in each subproblem (supernode), the records would form another table, as shown in Table 2. Each row can be viewed as a way to divide k into disjoint sets S_0 and S_1. For example, the first row represents the way of dividing k into $S_0 = \{0, 1, 2, 3\}$ and $S_1 = \{4, 5, 6, 7\}$. Besides, supernodes 01 and 10 are colored with colors A and B, respectively. Supernodes 00 and 11 are further divided recursively in the second row. Note that the coloring information for node aa, $0 \le a \le k-1$, is not shown in Table 2, since it is a special case.

Since each group of each row in Table 2 represented supernodes 01 and 10, colored with two colors, we further



(a) (b) (c)

Figure 2. The relationship between $dB(4,2)$ and $dB(2,2)$. (a) Connections between supernodes. (b) Connections within supernodes. (c) The directed de Bruijn graph $dB(2,2)$.

Table 3. The independence table extended from Table 2.

	0	1	2	3	4	5	6	7
A	L	L	L	L	R	R	R	R
B	R	R	R	R	L	L	L	L
C	L	L	R	R	L	L	R	R
D	R	R	L	L	R	R	L	L
E	L	R	L	R	L	R	L	R
F	R	L	R	L	R	L	R	L

extend each row into two rows, each for one supernode, as shown in Table 3, the *independence table*. Each supernode, which corresponds to one group in one row, consists the node set $\{v_2v_1|\ v_2$ is a digit marked with L, and v_1 is a digit marked as $R\}$. For example, the supernode in the fourth row, one supernode consists of {20, 21, 30, 31}, and the other supernode consists of {64, 65, 74, 75}. Still note that all nodes aa, $0 \le a \le k-1$, are not shown in Table 3.

In Table 3, each supernode forms an independent set. And furthermore, Table 3 reveals more information for independent sets. It is clear that the set $W_i = \{v_2v_1|\ v_2$ and v_1 are two digits marked with L and R, respectively, in row $i\ \}$ must be an independent set since no left (marked as L) digit is the same as a right (marked as R) digit. Thus, only one color is needed for coloring the nodes in one row. Note that W_i contains more nodes than the supernodes indicated in row i. With this observation, we can rearrange the markers L and R in each row such that each node, except node aa, appears in at least one row. This new rearrangement corresponds to a new coloring way.

Now we formally define the *independence table* $\mathcal{T}$ for $dB(k,2)$, which contains some rows and k columns, where each entry is of value either L or R. Each row r corresponds to the node set $\{ij|\mathcal{T}_{r,i} = L\ and\ \mathcal{T}_{r,j} = R\}$. A table $\mathcal{T}$ is *valid* if and only if for all nodes $ab \in dB(k,2), a \ne b$, there exists at lease one row r in $\mathcal{T}$ such that $\mathcal{T}_{r,a} = L$ and $\mathcal{T}_{r,b} = R$. The nodes aa, $0 \le a \le k-1$, are not illustrated in $\mathcal{T}$. Since the nodes in each row form an independent set, each row needs only one unique color. And one extra color is needed for nodes aa. By summing up above statement, we get a conclusion as follows.

Theorem 2. *The de Bruijn graph $dB(k,2)$ is $p+1$ colorable if there exists a valid independence table with p rows and k columns.*

Our job for coloring $dB(k,2)$ now becomes to design a valid independence table with minimum number of rows. Our algorithm cannot tell how many colors are required for $dB(k,2)$ directly. But for a given number of colors, we can tell the maximum value of k in $dB(k,2)$ we can reach. If we have $p+1$ colors, let each column correspond to a unique combination of p elements. To make it valid, we only use a subset of combinations, q elements chosen from p elements, whose size is $\binom{p}{q}$. To maximize k, we let $q = \lfloor p/2 \rfloor$. Thus we can use $p+1$ colors to color $dB(\binom{p}{\lfloor p/2 \rfloor}, 2)$. In other words, $dB(k,2)$ can be colored with $p+1$ colors where $\binom{p-1}{\lfloor (p-1)/2 \rfloor} < k \le \binom{p}{\lfloor p/2 \rfloor}$.

Let F be the color assignment function such that $F(v)$ is the color assigned to node v. Assume the color index begins at 0. Our algorithm is given as follows.

Algorithm 2. $p+1$ *Coloring Algorithm*

Input: The de Bruijn graph $dB(k,2)$.

Output: The coloring of $dB(k,2)$ with $p+1$ colors where $\binom{p-1}{\lfloor (p-1)/2 \rfloor} < k \le \binom{p}{\lfloor p/2 \rfloor}$

Step 1: Find p where $\binom{p-1}{\lfloor (p-1)/2 \rfloor} < k \le \binom{p}{\lfloor p/2 \rfloor}$

Step 2: For the node $v = v_2v_1$, where $v_2 = v_1$, let $F(v) = p$.

Step 3: For the node $v = v_2v_1$, $v_2 \ne v_1$, do the following.

Step 3.1: Find the v_2th and v_1th combinations of $\binom{p}{\lfloor p/2 \rfloor}$ in lexical order.

Step 3.2: Find the index r where the rth element of v_2th combination is L and rth element of v_1th combination is R. Let $F(v) = r$. If more than one such r exists, arbitrarily assign any one such r to $F(v)$.

Theorem 3. *The chromatic number of $dB(k, 2)$ is $p + 1$ when $k = \binom{p}{\lfloor p/2 \rfloor}$.*

4 Extension to $dB(k, m)$

Here we propose an extension scheme which allows us to color a $dB(k, m)$ graph, $m \geq 3$, based on any given valid coloring of $dB(k, 2)$ without requiring more colors.

Let $F(v)$ denote the color assigned to node $v = v_m v_{m-1} \ldots v_1$ in $dB(k, m)$. Let $G(x, y)$ be the color look-up function where $G(x, y)$ is the color of the node labeled as xy in the given colored $dB(k, 2)$. In $dB(k, m)$, for each node $v = v_m v_{m-1} \ldots v_1$, we use a 3-tuple (b_v, δ_v, a_v) to represent v, where $a_v = v_1$, $b_v = v_{\delta_v+1}$ is the rightmost digit such that $b_v \neq v_1$. In other words, a_v is the rightmost repeated digit and δ_v is the repeated length ($a_v = v_1 = v_2 = \ldots = v_{\delta_v}$). For example, node $\ldots 13333$ is encoded as the 3-tuple $(1, 4, 3)$.

Our algorithm for extending the coloring of $dB(k, 2)$ to $dB(k, m), m \geq 3$, is given as follows:

Algorithm 3. *Extending Algorithm*

Input: A valid coloring of $dB(k, 2)$ and an uncolored $dB(k, m)$.

Output: A valid coloring of $dB(k, m)$ with the same number of colors used in $dB(k, 2)$.

Step 1: For each node v, find (b_v, δ_v, a_v)

Step 2: For each node v,
If δ_v is odd, then $F(v) = G(b_v, a_v)$;
otherwise $F(v) = G(a_v, a_v)$;

For example, node $\ldots 2111$ is encoded as $(2, 3, 1)$ and thus colored as node 21 in $dB(k, 2)$ since its δ value is odd. Similarly, node $\ldots 13333$ is encoded as $(1, 4, 3)$ and colored as node 33 in $dB(k, 2)$.

Theorem 4. *Algorithm 3 assigns different colors to each pair of adjacent nodes in $dB(k, m), m \geq 3$.*

5. Conclusions and Future Works

In this paper, we first proposed two coloring algorithms to color the nodes in de Bruijn graph $dB(k, 2)$. One needs $2\lceil \log_2 k \rceil + 1$ colors and the other needs $p + 1$ only colors, where $\binom{p-1}{\lfloor (p-1)/2 \rfloor} < k \leq \binom{p}{\lfloor p/2 \rfloor}$. It is optimal for $dB(k, 2)$ when $k = \binom{p}{\lfloor p/2 \rfloor}$. We also proposed an extension algorithm to color $dB(k, m)$ for $m \geq 3$, based on a colored $dB(k, 2)$ without increasing the number of colors. By combining the extension and the second coloring algorithm, we solved the coloring problem on k-ary de Bruijn graphs with $p + 1$ colors.

However, we are not sure if the method, combining the second coloring algorithm and the extension algorithm, is optimal for the $dB(k, m)$ graphs, $m \geq 3$. Even if this method is optimal for $dB(k, 2)$, to find the optimal solution for the $dB(k, m)$ graph still remains an open problem. Related proofs are also worthy of further study.

References

[1] M. Beale and S. M. S. Lau. Complexity and auto-correlation properties of a class of de Bruijn sequence. *Electronics Letters*, 22(20):1046–1047, May 1986.

[2] J. C. Bermond and P. Fraigniaud. Broadcasting and gossiping in de Bruijn networks. *SIAM Journal on Computing*, 23(1):212–225, Feb. 1994.

[3] J. Bruck, R. Cypher, and C. T. Ho. Fault-fault de Bruijn and shuffle-exchange networks. *IEEE Transactions on Parallel and Distributed Systems*, 5(5):548–553, May 1994.

[4] A. H. Esfahanian and S. L. Hakimi. Fault-tolerant routing in de Bruijn communication networks. *IEEE Transactions on Computers*, C-34(9):777–788, Sept. 1985.

[5] M. G. Gouda and F. F. Haddix. The alternator. In *ICDCS '99: Workshop on Self-stabilizing Systems*, pages 48–53, Washington, DC, USA, 1999. IEEE Computer Society.

[6] S.-T. Huang and B.-W. Chen. Optimal 1-fair alternators. *Information Processing Letters*, 80(3):159–163, 2001.

[7] J.-W. Mao. *The Coloring and Routing Problems on de Bruijn Interconnection Networks*. Ph.D. Dissertation, National Sun Yat-Sen University, Kaohsiung, Taiwan, July 2003.

[8] J.-W. Mao and C.-B. Yang. Shortest path routing and fault-tolerant routing on de Bruijn networks. *Networks*, 35(3):207–215, 2000.

[9] J.-W. Mao and C.-B. Yang. A design for node coloring and 1-fair alternator on de Bruijn networks. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, volume 4, pages 1872–1878, Las Vegas, Nevada, USA, 2003.

[10] D. K. Pradhan and S. M. Reddy. A fault-tolerant communication architecture for distributed systems. *IEEE Transactions on Computers*, 31(9):863–870, Sept. 1982.

[11] M. A. Sridhar. The undirected de Bruijn graph: Fault tolerance and routing algorithms. *IEEE Transactions on Circuits and Systems I-Fundamental Theory and Applications*, 39(1):45–48, 1992.

[12] M. A. Sridhar and C. S. Raghavendra. Fault-tolerant networks based on the de Bruijn graph. *IEEE Transactions on Computers*, 40(10):1167–1174, Oct. 1991.